Optimal control of nonlinear hyperbolic systems on networks: gradient-based and deep learning approaches

Mickael Bestard

Thèse

INSTITUT DE RECHERCHE MATHÉMATIQUE AVANCÉE

UMR 7501

Strasbourg

présentée pour obtenir le grade de docteur de l'Université de Strasbourg
Spécialité MATHÉMATIQUES APPLIQUÉES

Mickaël Bestard

Optimal control of nonlinear hyperbolic systems on networks: gradient-based and deep Learning approaches

Soutenue le 15 décembre 2023
devant la commission d'examen

Carlotta Donadello, examinatrice
Nicolas Forcadel, rapporteur
Philippe Helluy, examinateur
Laurent Navoret, co-encadrant
Yannick Privat, directeur de thèse
Ulrich Razafison, rapporteur
Invités :
Emmanuel Franck, co-encadrant
Thibault Liard

https://irma.math.unistra.fr

cnrs

IRMA
Institut de Recherche Mathématique Avancée

Université de Strasbourg

Contents

Introduction en français

Les équations dérivées de la mécanique des fluides peuvent être utilisées pour modéliser une large variété de phénomènes physiques intervenant dans des situations très différentes. Au-delà même des études de comportements de liquides ou de gaz qui viennent immédiatement à l'esprit avec, par exemple, des applications en météorologie, climatologie ou en aérodynamisme, il est pertinent de noter qu'une très large classe de systèmes présente un comportement similaire lorsqu'ils sont étudiés à l'échelle macroscopique. L'un de leurs principaux points communs est certainement que, contrairement à ce que l'on peut observer par exemple avec des modèles paraboliques en mécanique du solide, ils ne propagent l'information qu'à une vitesse finie. Les exemples incluent bien sûr la propagation d'ondes électromagnétiques modélisées par les équations de Maxwell, mais également le cheminement de requêtes http à travers un réseau internet de routeurs interconnectés, ou l'évolution d'une épidémie entre différentes villes reliées par des routes. L'objectif principal de cette thèse est la simulation numérique et le contrôle optimal de ce type de système, en utilisant dans un premier temps diverses méthodes déterministes basées sur le gradient, puis l'apprentissage profond avec des réseaux de neurones pour réduire efficacement la taille du problème.

La première partie de ce manuscrit s'intéresse à un modèle hydrodynamique de trafic routier qui est une extension par Coclite et al.[23] du modèle de Lighthill-Whitham-Richards [61, 82] à un graphe représentant un réseau routier. La gestion des intersections d'un tel réseau constitue un défi autant du point de vue de la modélisation que du contrôle, à cause notamment du couplage fortement non linéaire qui relie des solutions très peu régulières (L^1). Par ailleurs, pour des raisons de persistance des données ainsi que de reproductibilité des résultats, le code associé aux simulations numérique est libre. Sa documentation fait l'objet du chapitre 3.

Dans un deuxième temps, nous nous intéresserons à un moyen d'accélérer les calculs, largement utilisé dans la recherche et l'industrie, les bases réduites [51]. Il s'agit d'une question cruciale, car les algorithmes d'optimisation et de contrôle peuvent avoir besoin d'usage intensif de simulations non linéaires, et il est impératif de rendre ces appels rapides tout en conservant une bonne fiabilité des résultats. Nous verrons que l'approche classique de réduction de modèle se heurte à la forte non-linéarité du système considéré, et proposerons une approche de réduction robuste utilisant un réseau de neurones pour évaluer le terme fortement non linéaire.

Le manuscrit se conclut par un projet réalisé en colaboration avec Leo Meyers et Florent Noisette sous la direction de Bastien Polizzi, Sébastian Minjeaud, Olivier Bernard et Thierry Goudon lors d'une école d'été. Ces travaux traitent eux-aussi de non-linéarité dans un modèle de mécanique des fluides, cette fois-ci appliqué à la biologie. Il s'agit en effet de simuler de façon précise l'évolution de micro-algues et de leur matrice extracellulaire dans un milieu aqueux. L'apport principal de ce chapitre est le développment d'un prédicteur-correcteur sur grille décalée permettant de restituer correctement le champ des vitesses malgré des couplages fortement non linéaires entre les phases ainsi que plusieurs forces et phénomènes biologiques comme la gravitation ou la

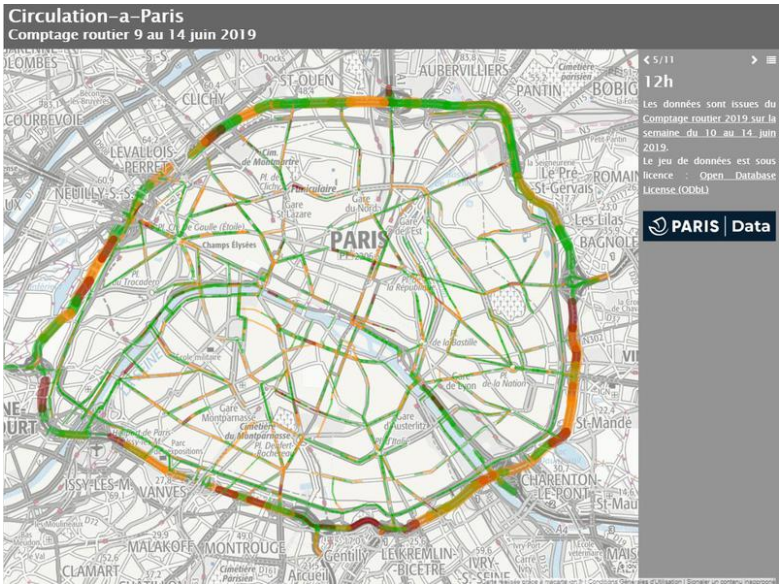


Figure 1: Macroscopic view of Paris' road network. https://www.data.gouv.fr/fr/reuses/circulation-a-paris/

photosynthèse.

Chapitre 1 : Revue de littérature et modélisation du problème de trafic sur graphe

Ce chapitre présente les différentes grandes catégories existantes de modèles de trafic routier avant d'introduire la modélisation macroscopique dont il est question dans cette thèse, avec notamment le couplage non linéaire aux intersections, et donne une vision générale des outils qui seront déployés par la suite pour formuler et analyser le problème de contrôle ainsi que son approximation numérique.

Le modèle LWR décrit l'évolution en espace et en temps d'une densité de véhicules ρ sur une route $[a, b] \subset \mathbb{R}$ soumise à un flux concave $f(\rho)$, avec une équation de conservation de la forme :

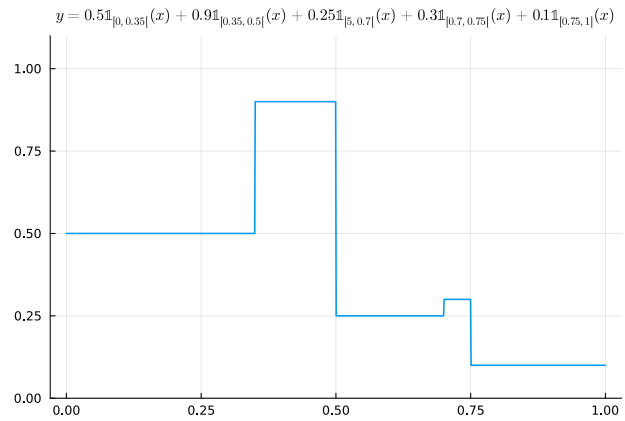
$$\partial_t \rho(t, x) + \partial_x f(\rho(t, x)) = 0, \quad (t, x) \in (0, T) \times [a, b]. \tag{1}$$

à laquelle on adjoint des conditions aux limites appropriées faisant intervenir des hypothèses de modélisation quant à la répartition des flux des véhicules aux jonctions à chaque instant.

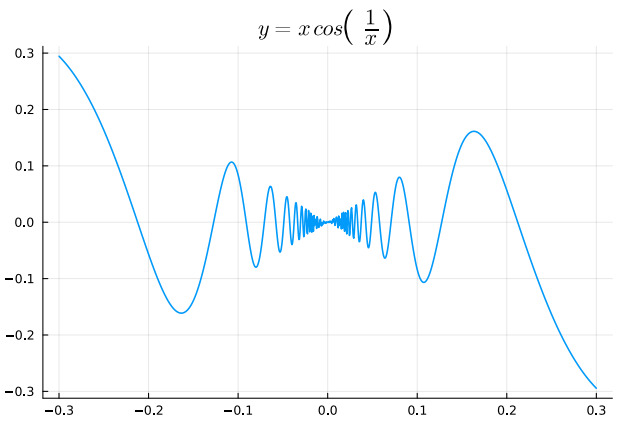
Un point important est la régularité et la compacité des solutions à variation bornée, cette notion, illustrée Figure 2, est prépondérante dans l'existence de solutions ou la convergence d'un algorithme itératif vers un contrôle minimisant effectivement le critère considéré.

Chapitre 2 : Scénario optimal d'une évacuation de routes dans un environement urbain

Comment désengorger une route du trafic routier le plus efficacement possible et en un temps donné, afin de permettre par exemple le passage de véhicules d'urgence ? La Figure 3 nous montre un exemple de situation

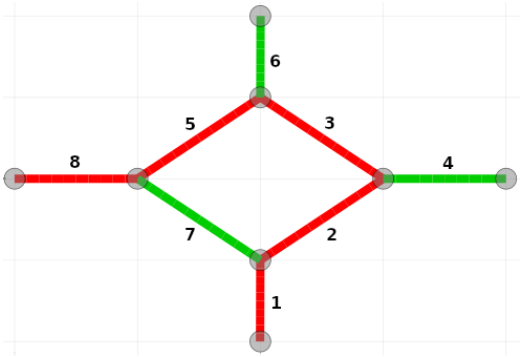


(a) Une fonction non continue à variation bornée : la fonction en escalier, souvent rencontrée lors de la modélisation d'ondes de choc.

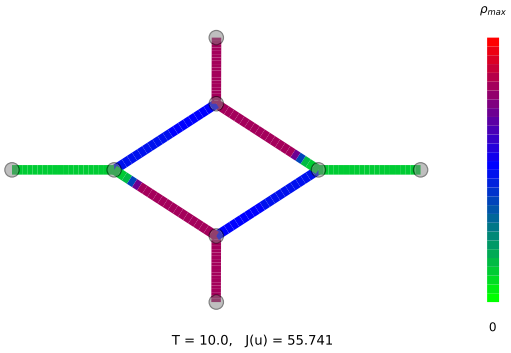


(b) Une fonction continue (par prolongement) qui n'est pas à variation bornée : $y(x) = xcos(1/x)$, avec des instabilités au voisinage de zéro.

Figure 2: Exemples de fonctions classiques bv et non bv.



(a) Road network and route to empty in red.



(b) Density at final time without control.

Figure 3: (Rond-point) Gauche : Configuration du rond-point. Les routes $(2, 3, 5, 7)$ du cercle sont dans le sens inverse des aiguilles d'une montre, les routes $(1, 6)$ sont entrantes et les routes $(4, 8)$ sont sortantes. Le trajet à vider $(1, 2, 3, 5, 8)$ est en rouge. A droite : Simulations numériques sans contrôle au temps $T = 10$.

défavorable dont nous chercherons à nous prémunir. Nous nous intéressons à cette question que nous reformulons comme un problème de contrôle optimal. Nous considérons un modèle macroscopique de trafic routier sur réseau, semi-discrétisé en espace et décidons de nous donner la possibilité de contrôler le flux à certaines jonctions. Notre objectif est de lisser et diminuer le trafic le long d'un chemin donné dans un temps donné. Une contrainte de parcimonie est imposée sur les contrôles, afin de s'assurer que les stratégies optimales sont réalisables en pratique.

On est donc ammenés, après discrétisation de (1) avec $N_c \in \mathbb{N}$ mailles sur chacune des $N_r \in \mathbb{N}$ routes :

$$\boldsymbol{\rho}(t) = \left(\rho_{i,j}(t)\right)_{\substack{1\le i\le N_r\\ 1\le j\le N_c}} = \left(\frac{1}{\triangle x_{i,j}}\int_{x_{i,j-\frac{1}{2}}}^{x_{i,j+\frac{1}{2}}} \rho(t,x)\,\mathrm{d}x\right)_{\substack{1\le i\le N_r\\ 1\le j\le N_c}}, \tag{2}$$

à minimiser une fonctionnelle du type :

$$\mathcal{J}(\mathbf{u}) = \sum_{i \in \text{roads}} \sum_{j \in \text{mesh}(i)} \rho_{i,j}(T; \mathbf{u}), \tag{3}$$

sous la contrainte dynamique donnée par le schéma aux volumes finis contrôlé suivant :

$$\begin{cases} \dfrac{\mathrm{d}\boldsymbol{\rho}}{\mathrm{d}t}(t) = f^{FV}\left(\boldsymbol{\rho}(t), \boldsymbol{\gamma}(t)\right), & t \in (0,T) \\ \boldsymbol{\gamma}(t) = \phi^{LP}\left(\boldsymbol{\rho}(t), \mathbf{u}(t)\right), & t \in (0,T) \\ \boldsymbol{\rho}(0) = \boldsymbol{\rho}_0, \end{cases} \tag{4}$$

où f^{FV} est le flux volumes finis et $\boldsymbol{\gamma}$ est le flux numérique aux jonctions. La fonction ϕ^{LP} introduite par Coclite et. al est modifiée dans ce manuscrit pour tenir compte du contrôle. On calcule la répartition optimale des flux aux jonctions à l'aide d'un problème de programmation linéaire sur la somme des traces des flux entrants. À titre d'exemple, voici son expression pour une jonction très simple ayant une route entrante et une route sortante :

$$\gamma_1(t) = \gamma_2(t) = \min\left(\gamma_1^{\text{demand}}(t), (1 - u_2(t))\gamma_2^{\text{supply}}(t)\right), \tag{5}$$

où $\boldsymbol{\gamma}(t) = (\gamma_1(t), \gamma_2(t))$ et $\gamma_1^{\text{demand}}(t)$, $\gamma_2^{\text{supply}}(t)$ sont respectivement les flux maximaux en entrée et en sortie de la jonction. Ils sont calculés à partir des densitées à l'instant t.

Remark 1. *Il est important de noter ici que la façon de définir la dépendance de ϕ^{LP} par rapport au contrôle* $\mathbf{u}$ *est un choix de modélisation crucial auquel il faut faire attention pour conserver le caractère bien posé du problème démontré dans [23], en particulier quant à la conservation de la densité de voitures à travers les jonctions.*

Nous effectuons ensuite une analyse du problème de contrôle optimal, prouvant l'existence d'un contrôle optimal et dérivant des conditions d'optimalité, que nous réécrivons sous la forme d'une unique équation fonctionnelle. Nous utilisons ensuite cette formulation pour dériver un algorithme mixte que nous interprétons comme un mélange de deux méthodes : une méthode de descente combinée avec une méthode de point fixe. Nous vérifions finalement par des expériences numériques l'efficacité de cette méthode sur des exemples de graphes, d'abord simples, puis plus complexes, avant de mettre en évidence l'efficacité de notre approche en la comparant aux méthodes standard.

Chapitre 3 : Un code Julia pour le contrôle optimal d'un réseau routier

Dans un souci de reproductibilité des résultats et pour rendre les travaux du précédent chapitre réutilisables, nous mettons le code source à disposition et présentons son utilisation. Ce programme est codé en Julia et peut être trouvé en accès libre en suivant ce lien :

https://github.com/mickaelbestard/TRoN.jl

Le chapitre commence par la présentation de la structure générale du code ainsi que l'interface permettant d'interagir avec, que ce soit pour reproduire les résultats ou le réutiliser pour d'autres problèmes faisant intervenir

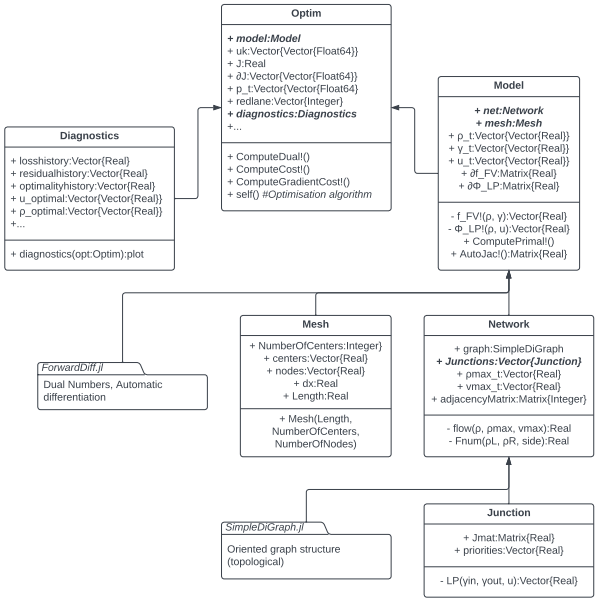


Figure 4: Global structure of the Julia code for optimal control of traffic flow.

une structure similaire. La fin du chapitre entre un peu plus dans les détails propres au langage de programmation utilisé pour présenter des perspectives d'amélioration du code du point de vue de son empreinte mémoire et de sa rapidité, deux sujets particulièrement reliés en Julia.

Chapitre 4 : Outils pour les modèles réduits et le machine learning

La décomposition en modes propres permet de capturer la structure sous-jacente à un jeu de données et de filtrer les informations redondantes. Cette technique de compression est efficace pour reformuler un système d'équations de grande dimension $N >> 1$ tel que :

$$\frac{\mathrm{d}\boldsymbol{\rho}(t)}{\mathrm{d}t} = F(\boldsymbol{\rho}(t)), \quad \boldsymbol{\rho}(t) \in \mathbb{R}^N, \tag{6}$$

en un système tronqué de taille réduite $m << N$:

$$\frac{\mathrm{d}\hat{\boldsymbol{\rho}}(t)}{\mathrm{d}t} = \hat{F}(\hat{\boldsymbol{\rho}}(t)), \quad \hat{\boldsymbol{\rho}}(t) \in \mathbb{R}^m, \tag{7}$$

où $\hat{\rho}$ est obtenu par projection (compression) de ρ sur une base judicieusement choisie pour garantir la fiabilité des données réduites. Typiquement, nous travaillerons avec $N = 800$ et $m \leq 20$ et noterons $\Phi \in \mathbb{R}^{N\times n}$ de sorte que :

$$\hat{\boldsymbol{\rho}} = \Phi^T \boldsymbol{\rho}. \tag{8}$$

Cette méthode dite d'interpolation empirique (EIM), qui existe aussi dans sa version dynamique (DEIM) pour les cas faiblement non linéaires, souffre néanmoins grandement de la forte non-linéarité du second membre du système.

Nous choisissons d'y remédier à l'aide de réseaux de neurones. Après en avoir exposé brièvement les

principaux aspects, nous rappelons une version de la descente de gradient qui permet de calculer de façon efficace l'erreur de prédiction du réseau par rapport à un grand jeu de données et qui est la base des méthodes utilisées dans les librairies standards de machine learning comme Adam ou RMS prop [53]: la descente de gradient stochastique [83].

Chapitre 5 : Approches comparées d'hyper-réduction par apprentissage profond

Nous utilisons les bases réduites pour approcher une famille d'équations différentielles ordinaires non linéaires, indexée par un paramètre du modèle que l'on notera μ :

$$\frac{\mathrm{d}\boldsymbol{\rho}_\mu(t)}{\mathrm{d}t} = F(\boldsymbol{\rho}_\mu(t)), \quad \boldsymbol{\rho}_\mu(t) \in \mathbb{R}^N. \tag{9}$$

Dans le chapitre 5, nous traiterons spécifiquement du cas où le modèle est donné par (4) sur une route à une entrée et une sortie, avec pour paramètre la densité maximale de voiture autorisée sur la route sortante. Le comportement de la solution numérique pour différentes valeurs du paramètre est illustré Figure 5, et l'on remarque que les basses valeurs du paramètre favorisent l'apparition de chocs tandis que les hautes valeurs génèrent des ondes de raréfactions.

En supposant que l'on puisse déduire $\boldsymbol{\rho}_\mu$ d'une représentation réduite $\hat{\boldsymbol{\rho}}_\mu \in \mathbb{R}^m$ de sorte que $\boldsymbol{\rho}_\mu \approx \Phi\hat{\boldsymbol{\rho}}_\mu$, la

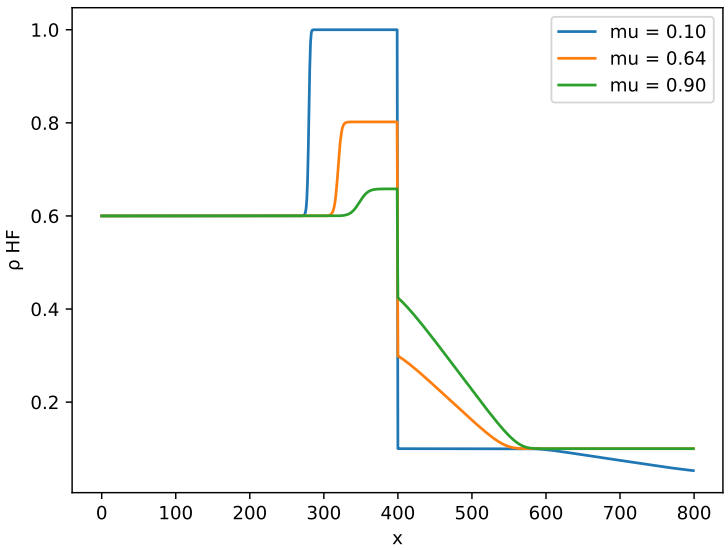


Figure 5: Densités de véhicules au temps final pour différentes valeurs de la densité maximale sur la deuxième route. Discrétisation spatiale : $N_c = 400$ cellules par route.

non-linéarité annulerait l'intérêt de l'utilisation d'un modèle réduit. Nous serions en effet ammenés à résoudre :

$$\Phi^T \frac{\mathrm{d}\boldsymbol{\rho}_\mu(t)}{\mathrm{d}t} = \frac{\mathrm{d}\hat{\boldsymbol{\rho}}_\mu(t)}{\mathrm{d}t} = \Phi^T F(\Phi\hat{\boldsymbol{\rho}}_\mu(t)), \quad \boldsymbol{\rho}_\mu(t) \in \mathbb{R}^N,\ \hat{\boldsymbol{\rho}}_\mu(t) \in \mathbb{R}^m, \tag{10}$$

impliquant une évaluation du flux F en grande dimension.

Nous formons alors un réseau de neurones, paramétré par θ, dont le but sera d'apprendre à évaluer le flux

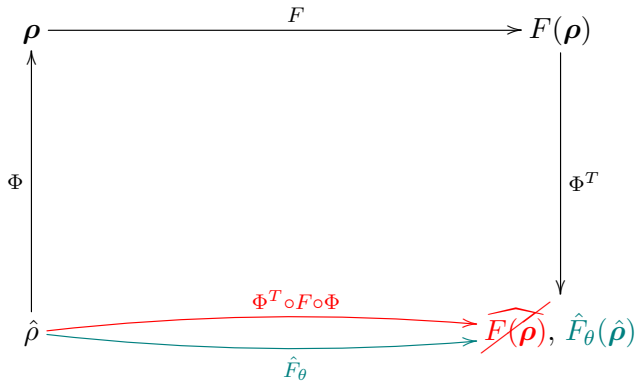


Figure 6: Objectif schématisé de la méthode d'hyper-réduction par apprentissage profond.

fortement non linéaire en n'utilisant que la base réduite (flèche bleue sur la Figure 6), c'est-à-dire de sorte que le calcul ne fasse pas intervenir la grande dimension du problème initial (flèche rouge sur la Figure 6).

Les deux approches utilisent le même réseau de neurones $\hat{F}_\theta$, un jeu de données $\hat{\boldsymbol{\rho}}_\mu(t)$ pour certaines valeurs de (t, μ) et ne diffèrent que par leur fonction de perte. Dans le premier cas, appelé "Neural Closure" (NC), nous cherchons à minimiser l'erreur entre le second membre prédit par le réseau de neurones $\hat{F}_\theta(\hat{\boldsymbol{\rho}}_\mu, \mu)$ et la compression $\Phi^T(\cdot)$ du second membre provenant du modèle de grande dimension $F(\hat{\boldsymbol{\rho}}_\mu; \mu)$:

$$\mathcal{L}^{NC}(\theta) = \text{error}\left(\hat{F}_\theta(\hat{\boldsymbol{\rho}}_\mu, \mu), \Phi^T F(\hat{\boldsymbol{\rho}}_\mu; \mu)\right). \tag{11}$$

Dans la seconde approche, appelée "Differentiable Physics" (DP), nous notons :

$$S^1_\theta(\hat{\boldsymbol{\rho}}_\mu(t_n)) := \hat{\boldsymbol{\rho}}_\mu(t_n) + \Delta t \hat{F}_\theta(\hat{\boldsymbol{\rho}}_\mu(t_n), \mu) \tag{12}$$

l'intégrateur en temps construit en utilisant le réseau de neurones comme flux d'intégration, et écrivons S^K_θ les composées itérées de S^1_θ nous transportant l'état réduit d'un instant t_n à un instant t_{n+K}.

Nous souhaitons ainsi minimiser l'erreur commise par cet intégrateur sur K pas de temps en le comparant à la compression de la trajectoire de référence $\Phi^T \boldsymbol{\rho}_\mu(t_n), \cdots, \Phi^T \boldsymbol{\rho}_\mu(t_{n+K})$:

$$\mathcal{L}^{DP}_K(\theta) = \text{error}(S^K_\theta(\hat{\boldsymbol{\rho}}_\mu(\cdot)), \hat{\boldsymbol{\rho}}_\mu(\cdot + K)). \tag{13}$$

C'est avec cette dernière approche que nous obtenons les résultats les plus prometteurs.

Chapitre 6 : Schémas numériques pour les modèles de théorie des mélanges avec contrainte de remplissage : application aux écosystèmes de biofilms

Le dernier chapitre traite de l'évolution de micro-algues dans un milieu aqueux, modélisée par un mélange de trois phases. La complexité du modèle provient de la non-linéarité des termes d'advection, des couplages, ainsi que des phénomènes physiques (gravitation, pression) et biologiques (photosynthèse) pris en compte. La stratégie numérique que nous avons utilisée repose sur l'utilisation de grilles décalées pour améliorer l'évaluation des vitesses aux interfaces des mailles, ainsi que d'un prédicteur-correcteur utilisant le gradient de la pression pour

corriger l'erreur commise sur le calcul des vitesses.

Il s'agit d'une prépublication issue d'un projet réalisé lors du Centre d'Eté Mathématique de Recherche Avancée en Calcul Scientifique (CEMRACS) en 2022 avec Léo Meyer et Florent Noisette, sous la direction de Bastien Polizzi, Sébastian Minjeaud, Olivier Bernard et Thierry Goudon. Ce projet s'insère naturellement dans le contexte général de ce manuscrit, car il vise à trouver des solutions numériques pour résoudre le problème de la forte non-linéarité dans un modèle fluide.

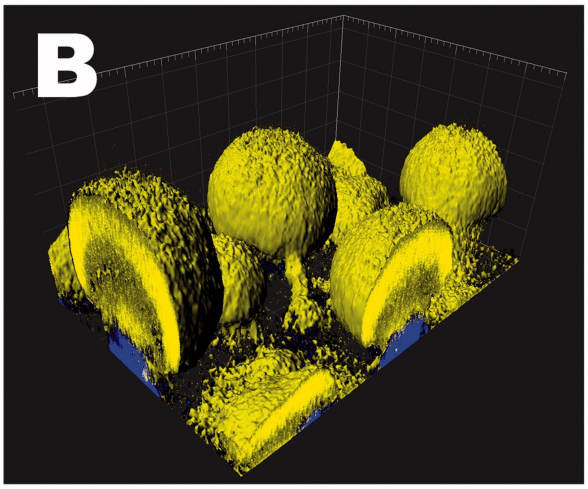


Figure 7: Vue macroscopique de micro-algues. Azeredo, Joana et al. "Critical review on biofilm methods." Critical reviews in microbiology vol. 43,3 (2017): 313-351.

Contributions et communications

Cette thèse a donné lieu aux contributions et communications suivantes :

- Articles :
  - *Optimal scenario for road evacuation in an urban environment*, preprint HAL (2023)
  - *Comparative approaches to hyper-reduction using deep learning*, article en cours de rédaction
  - *Numerical schemes for mixture theory models with filling constraint: application to biofilm ecosystems*. Ce papier est soumis, et sera publié dans le journal ESAIM: Proceedings and Surveys.
- Orateur invité :
  - *Simulation numérique avec Julia*, Séminaire INFOMATHS. Sorbonne Université et Université Paris-Dauphine (2022)
  - *Optimal scenario for road evacuation in an urban environment*, Workshop on Mathematical modeling, Analysis and Approximation of Vehicular and Pedestrian Traffic. Tour (2023)
- Conférences Nationales et Internationales :
  - *Simulation numérique avec Julia*, Congrès SMAI, La Grande Motte (2021)
  - CANUM, Evian (2022)
  - International Conference on Hyperbolic Problems, Malaga (2022)

- Séminaires :
  - Séminaire EDP, IRMA Strasbourg (2021)
  - Séminaire doctorants, IRMA Strasbourg (2021, 2022, 2023)
  - Rencontres doctorants "PhD Aways Days", Strasbourg, Luxembourg (2022)
  - Groupe de travail Machine Learning, *Stochastic gradient descent*, IRMA Strasbourg (2022)
- Vulgarisation scientifique :
  - Finaliste Alsace *Ma thèse en 180 secondes*, Strasbourg (2022)

Part I

Optimal Control Applied to Road Networks

Chapter 1

Tools for traffic modeling and analysis

This chapter first outlines road traffic model categories, then delves into macroscopic modeling, particularly nonlinear intersection coupling. It also previews the tools for analyzing the control problem and its numerical approximation. Emphasizing the significance of solutions' regularity and bounded variation, which plays a vital role in solution existence and iterative algorithm convergence toward optimal control.

1.1 Overview of existing traffic models

Depending on the specific scenario being studied, various approaches exist for modeling the phenomenon under consideration. These approaches vary in scale and the level of accuracy they aim to achieve.

At the microscopic scale, the behavior of individual vehicles can be described using Lagrangian microscopic models. These models focus on the trajectories, including speed and position, of each vehicle. Examples of such models include car-following models, as referenced in works such as [79, 43, 62], as well as cellular automaton models, as discussed in [54, 11, 50]. Additionally, kinetic traffic models, introduced in the 1960s, take a mesoscopic approach, representing vehicles as distributions in position-velocity space. These models have been explored in works such as [77, 26, 31, 38, 66, 71, 94].

Moving up to a macroscopic scale, the flow of vehicles is treated as a continuous medium. One notable macroscopic model is the one introduced by Lighthill, Whitham, and Richards (commonly denoted as LWR) in the 1950s, as documented in [61, 82]. In the subsequent sections of this work, we will primarily use the LWR model, as its continuous representation of flow lends itself well to formulating dynamic programming problems.

It's worth noting that the original first-order model from the 1950s has been extended to second-order models resembling Euler-type systems with viscosity, as discussed in [72, 95, 5, 63]. These extensions provide a more comprehensive understanding of traffic dynamics.

For a comprehensive introduction to this topic, see [42].

1.2 Macroscopic modeling of road traffic on a network

To study road traffic on a macroscopic scale, we use a continuous model introduced by Lighthill, Whitham, and Richards in 1955/1956, which describes the evolution of car density. While this model is very well known when applied to a single road, its generalization to networks raises theoretical and numerical difficulties that have given

rise to numerous studies. The junction problem is of central importance, since it is necessary to make sense of solutions that are a priori multivalued, as well as to properly model the distribution of vehicles from one road to another. This section starts with a formal derivation of the model to give an intuition of the underlying physics. Then, classical results of the domain are presented.

We write $\rho(t,x)$ the number of cars per unit of length at position $x \in \mathbb{R}$ and time $t \geq 0$. This quantity allows us to define the mass on any interval $I := [a,b]$ as being:

$$m(t) = \int_a^b \rho(t,x)\,dx. \tag{1.1}$$

This is a conserved quantity of the system, which means that:

$$m'(t) = \int_a^b \partial_t \rho(t,x)\,dx = 0. \tag{1.2}$$

The flow of vehicles, denoted as $f(\rho(t,x))$, is the mass of vehicles that can pass through x at time t. Assuming no source or loss of vehicles, its variations at the boundaries of a road give us the variation of the mass within it, i.e.

$$\begin{aligned} m'(t) = \int_a^b \partial_t \rho(t,x)\,dx &= \textit{ingoing flux} \;-\; \textit{outoing flux} \\ &= f(\rho(t,a)) - f(\rho(t,b)) \\ &= -\int_a^b \partial_x f(\rho(t,x))\,dx. \end{aligned}$$

We therefore obtain for some $T \geq 0$ the following scalar conservation law:

$$\partial_t \rho(t,x) + \partial_x f(\rho(t,x)) = 0, \quad (t,x) \in (0,T) \times [a,b]. \tag{1.3}$$

In [Lighthill, Whitham, Richards, 1955], the flux is given more specifically by:

$$f(\rho) = \rho v(\rho) = v_{\max}\rho(1 - \frac{\rho}{\rho_{\max}}), \tag{1.4}$$

with $v_{\max}$ and $\rho^{\max}$ the maximum speed and density allowed on the road. This choice is motivated by the fact that $v(0) = v^{\max}$ and $v(\rho^{\max}) = 0$. We drive indeed faster on an empty road than during a traffic jam, as illustrated in Figure 1.1.

Since this type of model is well known to produce discontinuous solutions in finite time, even for smooth initial data, we therefore need to weaken the notion of solution.

Definition 1 (Weak solution on a road)**.** *We say that ρ is a weak solution on road I if for any $\varphi : \mathbb{R}_{\geq 0} \times I \to \mathbb{R}$ smooth with compact support on $\mathbb{R}_{>0} \times \mathring{I}$, we have:*

$$\int_0^{+\infty} \int_I (\rho \partial_t \varphi + f(\rho)\partial_x \varphi)\,dxdt = 0. \tag{1.5}$$

For the associated Cauchy problem to be well-posed, the search for weak solutions is not yet sufficient to guarantee uniqueness. The family of functions we need is the weak entropic ones, in other words, those that are

physically relevant. These considerations are very standard and are discussed in detail in Coclite et al. [41].

On a network

A road network of N_r roads and N_v junctions is modelled by a directed graph $\mathcal{G} = (V, E)$ whose edges

$$E = \{I_i := [a_i, b_i] \subset \mathbb{R} \mid 1 \leq i \leq N_r\},$$

represent the roads and the vertices

$$V = \{J_k = \left((i_1^k, \cdots, i_{n^k}^k), (j_1^k, \cdots, j_{m^k}^k)\right) \mid 1 \leq k \leq N_v\},$$

the junctions, each junction J_k having n^k incoming and m^k outgoing roads.

The LWR model is straightforwardly extended to describe the behaviour of the vehicles densities $(\rho_i)_i$ on roads I up to a time $T \geq 0$, which readily yields the following Cauchy problem on the network:

$$\begin{cases} \partial_t \rho_i(t,x) + \partial_x f_i(\rho_i(t,x)) = 0, & (t,x) \in [0,T] \times [a_i, b_i], \\ \rho_i(0,x) = \rho_{i,0}(x), & x \in [a_i, b_i], \end{cases} \tag{1.6}$$

where flow and speed are given by:

$$f_i(\rho_i) = \rho_i v_i(\rho_i), \quad v_i(\rho_i) = v_i^{\max}\left(1 - \frac{\rho_i}{\rho_i^{\max}}\right), \tag{1.7}$$

with $v_i^{max} \geq 0$ the maximum speed allowed by the model on the road i and ρ_i being bounded by a given $\rho_i^{\max} \geq 0$. See Figure 1.1.

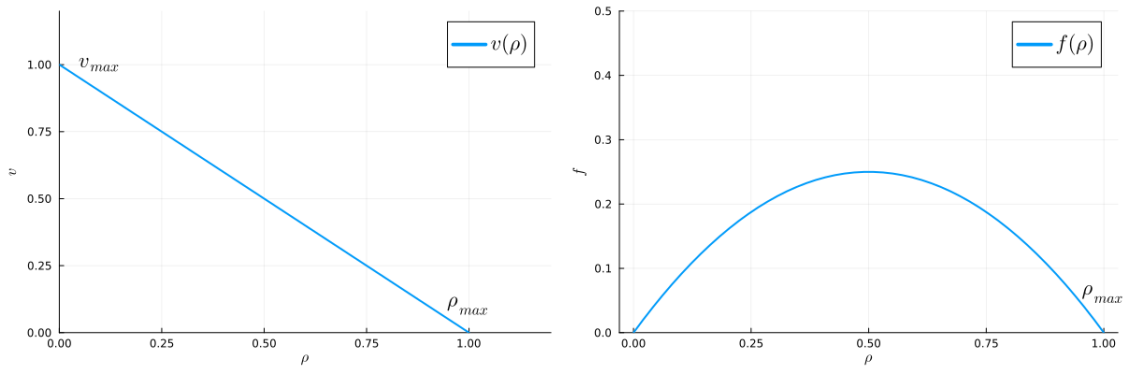


Figure 1.1: Local vehicles speed and flow in the LWR model, with $\rho^{\max} = v^{\max} = 1$.

Vehicles distribution at junctions

Defining the coupling of the roads at junctions in an appropriate way is however a non-trivial problem, since the density becomes multivaluated with as many crossing characteristics as the arity of the intersection. This has given rise to numerous works [REF], and we will follow the approach of *Coclite et al.* [23] by linking the solution at junctions from its flow traces.

We write for each time $t \in (0, T)$ the flow traces $\boldsymbol{\gamma}(t) = \left(\boldsymbol{\gamma}^L(t), \boldsymbol{\gamma}^R(t)\right)$ of the flux at boundaries, such that:

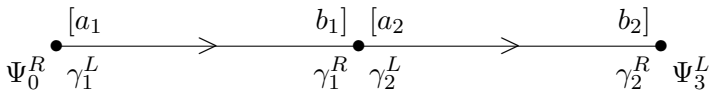


Figure 1.2: Junction with one ingoing and one outoing road.

- $\gamma_i^R(t) := f_i\left(\rho_i(t, b_i^-)\right)$ is the flow going **in** the junction **from** the road i, located at the right[1] of the ingoing road.
- $\gamma_j^L(t) := f_j\left(\rho_j(t, a_j^+)\right)$ is the flow going **from** the junction **to** the road j, located at the left of the outgoing road.
- $(\Psi_k(t))_k$ denote the flows at the junctions located next to graph's leaves, using ghost cells convention.

We provide an illustration of these notations in Figure 1.2, on a junction between one ingoing and one outoing road.

Considering a generic junction with n ingoing and m outgoing roads, *Coclite et al.* introduce the relation

$$\sum_{i=1}^{n} f_i(\rho_i(t, b_i)) = \sum_{j=n+1}^{n+m} f_j(\rho_j(t, a_j)), \tag{1.8}$$

to ensure the conservation of the flux. This means that the number of cars entering the intersection is the same as the number leaving it. But it does not determine uniquely where they come from or where they are going at if there are several incoming or outgoing roads (i.e. for $m, n \geq 2$). They therefore add constaints to the model such as assuming that there is a raw-stochastic distribution matrix $A \in \mathbb{R}^{m \times n}$ that links the input of a junction $(\gamma_i^R)_{1 \leq i \leq n}$ to its output $(\gamma_j^L)_{n+1 \leq j \leq n+m} = A(\gamma_i^R)_{1 \leq i \leq n}$. We then write

$$A := (\alpha_{ji})_{\substack{n+1 \leq j \leq n+m \\ 1 \leq i \leq n}}, \ 0 < \alpha_{ji} < 1, \ \sum_{j=n+1}^{n+m} \alpha_{ji} = 1, \tag{1.9}$$

with α_{ji} being the percentage of drivers arriving from the i-th road that take the j-th outgoing road.

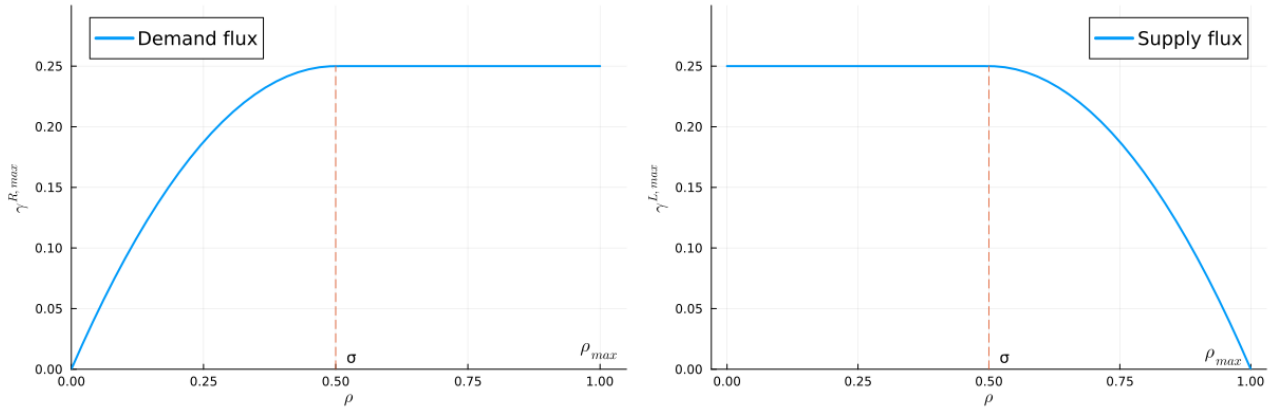
Finally, the main modelling assumption made to achieve coupling between roads is that drivers behave in such a way as to maximize flow, while respecting the intrinsic restrictions imposed, for example, by road size or speed limits. As a result, the flow at junctions will be an optimal compromise between maximum demand satisfaction and available supplies. These restrictions are computed as follows: by f_i concavity, there indeed exists a unique $\sigma_i \in [0, \rho_i^{\max}]$ such that $f_i'(\sigma_i) = 0$. In this point, f_i reaches its maximum so we define $(\gamma_i^{R,\max})_{1 \leq i \leq n}$ the maximum flows that can be obtained on incoming roads, i.e. the *demand flux*, Figure 1.3a, as:

$$\gamma_i^{R,\max} = \begin{cases} f_i(\rho_i(t, b_i)), & \text{if } \rho_i(t, b_i) \in [0, \sigma_i], \\ f_i(\sigma_i), & \text{if } \rho_i(t, b_i) \in [\sigma_i, \rho_i^{\max}], \end{cases} = f_i(\min\{\sigma_i, \rho_i(t, b_i)\}). \tag{1.10}$$

We define the same way the *supply flux* $(\gamma_j^{L,\max})_{n+1 \leq j \leq n+m}$ for the outgoing roads, see Figure 1.3b:

$$\gamma_j^{L,\max} = \begin{cases} f_j(\sigma_j), & \text{if } \rho_j(t, a_j) \in [0, \sigma_j], \\ f_j(\rho_j(t, a_j)), & \text{if } \rho_j(t, a_j) \in [\sigma_j, \rho_j^{\max}]. \end{cases} = f_j(\max\{\sigma_j, \rho_j(t, a_j)\}). \tag{1.11}$$

[1] the graph being oriented, we define "left" and "right" canonically from the direction of each road.



(a) demand flux **(b)** supply flux

Figure 1.3: Demand and supply fluxes for $\rho^{max} = v^{max} = 1$.

In order to write the optimal flow repartition, let us introduce the pavements:

$$\begin{aligned} \Omega_i &:= [0, \gamma_i^{R,\max}], \quad 1 \leq i \leq n, \\ \Omega_j &:= [0, \gamma_j^{L,\max}], \quad n+1 \leq j \leq n+m, \end{aligned} \tag{1.12}$$

which are used to define the following polytope:

$$\Omega := \{\boldsymbol{\gamma} := (\gamma_1, \cdots, \gamma_n) \in \Omega_1 \times \cdots \times \Omega_n \mid A\boldsymbol{\gamma} \in \Omega_{n+1} \times \cdots \times \Omega_{n+m}\}. \tag{1.13}$$

The optimal trade-off principle stated earlier then leads to solve, at **every junction** and for **each time**, the following linear programming:

$$\max_{\boldsymbol{\gamma} \in \Omega} \mathbf{1} \cdot \boldsymbol{\gamma}, \tag{1.14}$$

where $\mathbf{1} := (1, \cdots, 1) \in \mathbb{R}^n$.

See again *Coclite et al.* [23] for the definition of weak entropic solutions at junctions.

Having defined the notion of solution on each roads and junctions, we are now able to define the solution on an entire network.

Definition 2 (Admissible solution on a network)**.** *Given initial datum $\bar{\rho}_i : I_i \to \mathbb{R}$ and boudary conditions $\psi_i : [0, +\infty[\to \mathbb{R}$ functions of L^∞, we say that a collection of functions $\rho = (\rho_1, \cdots, \rho_{N_r})$ with $\rho_i : [0, +\infty[\times I_i \to \mathbb{R}$ continuous as functions from $[0, +\infty[$ into L^1_{loc}, is an admissible solution if ρ_i is a weak entropic solution to (1.6) on I_i, and such that at each junction ρ is a weak solution and is an admissible weak solution in case of bounded variations.*

Note that we're working in the context of bounded variations. This notion has several implications which will be discussed in section 1.3. For now, let's quickly recall the notion of Riemann problem, and how to solve it in this context of networks. This is interesting both for theoretical reasons with the existence of a solution to the Cauchy problem on a network by the so-called *wavefront tracking* algorithm [23], and for numerical reasons since it allows to construct finite volume schemes.

1.2.1 Analysis of the Riemann problem

The existence and uniqueness for the Cauchy problem on network are established in [23] using the wavefront tracking algorithm, which relies on approximating the solution by piecewise constant functions and solving each local problem using the fact that in hyberbolic dynamics, informations travels at finite speed. We will first recall Riemann problems on single roads before giving the extensions on junctions and networks.

Riemann problem at roads

We aim to solve for each road the following kind of Cauchy problem, with Heavyside-like initial condition:

$$\begin{cases} \partial_t \rho(t,x) + \partial_x f(\rho(t,x)) = 0, \\ \rho(0,x) = \rho_0(x) = \begin{cases} \rho_L, & x < 0 \\ \rho_R, & x > 0 \end{cases} \end{cases} \tag{1.15}$$

This is a very well known problem, see [13] for instance, and the solution is given in the following theorem.

Theorem 1. *If $\rho_L < \rho_R$ then $f'(\rho_L) > f'(\rho_R)$ and we have an **entropic shock** propagating at the speed σ given by the Rankine-Hugoniot relation:*

$$\sigma = \frac{f(\rho_L) - f(\rho_R)}{\rho_L - \rho_R} = v_{\max}\left(1 - \frac{\rho_L + \rho_R}{\rho_{\max}}\right), \tag{1.16}$$

$$\rho(t,x) = r(\xi) = \begin{cases} \rho_L, & \xi < \sigma, \\ \rho_R, & \xi > \sigma. \end{cases} \tag{1.17}$$

*Conversely, if $\rho_L > \rho_R$ then $f'(\rho_L) < f'(\rho_R)$ and we are in the case of a **rarefaction wave**, given by:*

$$\rho(t,x) = r(\xi) = \begin{cases} \rho_L, & \xi < f'(\rho_L), \\ g(\xi), & f'\rho_L < \xi < f'(\rho_R), \\ \rho_R, & f'(\rho_R) < \xi, \end{cases} \tag{1.18}$$

where $g(\xi) = \frac{\rho_{\max}}{2}\left(1 - \frac{\xi}{v_{\max}}\right)$ is such that $f'(g(\xi)) = \xi$.

Riemann problem at junction

The Riemann problem at a junction $\mathcal{J} := ((I_1, \cdots, I_n), (I_{n+1}, \cdots, I_{n+m}))$ is given by Equation 1.6 on each road $(I_k)_{1 \le k \le n+m}$ with respect to the optimal coupling constraints from Equation 1.14, where the initial condition is a constant density on each road. According to the aformentioned modeling hypothesis, there exists a unique weak solution of this problem in the sense given by [[23], Theorem 3.1].

1.2.2 Cauchy problem on network

Using the wavefront tracking strategy developed in [29], knowledge of the Riemann problem is sufficient to obtain unique solutions to generic Cauchy problems. In fact, the idea of the wavefront tracking algorithm is

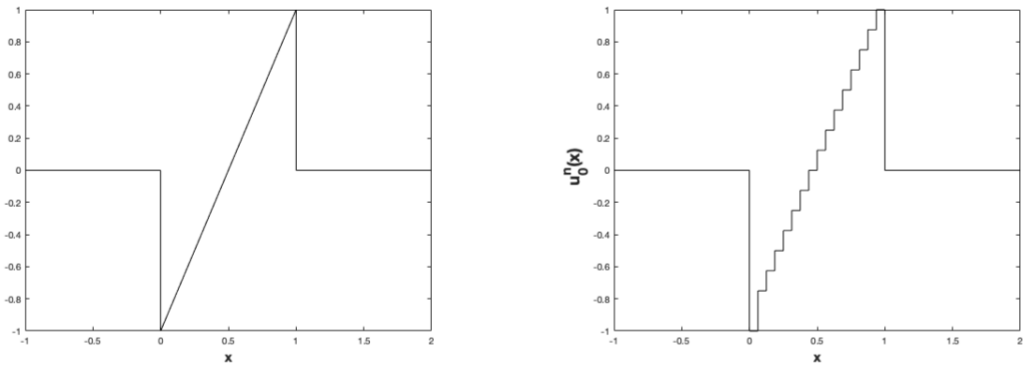


Figure 1.4: Rarefaction wave and its spatial discretization on a uniform mesh. T.Liard and E. Zuazua, 2023 [60]

to approximate the solution by a family of piecewise constant functions, solve the Riemann problems at each interface and then pass to the limit with the appropriate topology. It is important to note that uniqueness is only given in the following sense:

The solution given by the convergence of the wavefront tracking algorithm does not depend on the choice of the piecewise constant approximations.

However, there is no guarantee that another type of strategy will give the same solution. As far as we known, the latter problem remains open.

Definition 3 (Wavefront tracking approximation)**.** *The wavefront tracking algorithm reads as follows:*

- *let $\bar{\rho}$ be a piecewise constant map defined on the network*
- *we solve riemann problems on each road at the discontinuity points of $\bar{\rho}$ as well as the riemann problem at junctions from the traces of $\bar{\rho}$*
- *for each riemann problem,*
  - *if the obtained solution is a shockwave, do nothing*
  - *if the obtained solution is a rarefaction, we spatially discretize it in order to obtain discontinuities travelling at Rankine-Hugoniot speed. We thus deal with shockwaves only. An illustration of this step is reproduced Figure 1.4 and comes from [60]*
- *we obtained a new piecewise constant solution, to wich we apply the sames steps as above.*

Theorem 2 (Wavefront tracking approximation property)**.** *Let ρ_ν be a wavefront tracking approximation of Equation 1.3 with grid size given by ν. If this squence converges in L^1_{loc} when $\nu \to 0$, then the limit is a weak entropic solution of Equation 1.3.*

Theorem 3 (Coclite, Piccoli 2005)**.** *Let f be the flow from the LWR model, and consider a network in which each junction has at most two ingoing and two outoing roads. Let $\bar{\rho} \in L^1_{loc}$ be an initial datum. Then, for any time $T > 0$, we have the following properties:*

- *All approximating sequences generated by the wave-front tracking algorithm yields a unique solution ρ to System (1.15) on $[0,T]$, such that $\rho(0,\cdot)=\bar{\rho}$, obtained as the limit of piecewise-constant solutions whose flows derivatives never cancel.*
- *If the initial datum $\bar{\rho}$ belongs to $L^1(0,T)$ then $\rho(t,\cdot)$ belongs to L^1.*
- *Let $\rho(t,\cdot),\rho'(t,\cdot)\in L^1$ for a.e. time $t\geq 0$, both obtained by wavefront tracking approximation. Then:*

$$\left\|\rho(t,\cdot)-\rho'(t,\cdot)\right\|_{L^1}\leq\left\|\rho(0,\cdot)-\rho'(0,\cdot)\right\|_{L^1}.$$

1.3 About BV spaces

We introduce now the space of functions with bounded variation, which is important both for the existence of the Cauchy problem for the LWR model and the existence of an optimal control. Given a function f belonging to $L^1([0,T])$, its total variation on $[0,T]$ is defined as

$$\mathrm{TV}(f)=\sup\left\{\int_0^T f(t)\phi'(t)\,dt,\ \phi\in C_c^1([0,T]),\ \|\phi\|_{L^\infty([0,T])}\leq 1\right\}.$$

We denote by $BV(0,T)$ the space of functions of bounded variations,

$$BV(0,T):=\left\{f\in L^1(0,T), TV(f)<+\infty\right\},$$

and, endowed with the norm

$$\|\cdot\|_{BV(0,T)}=TV(\cdot)+\|\cdot\|_{L^1(0,T)},$$

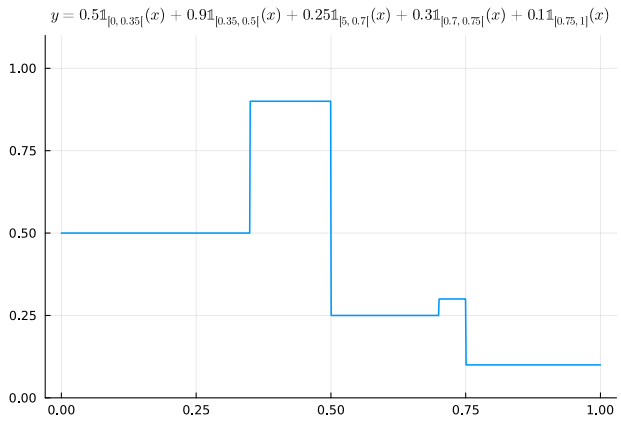
the space $\left(BV(0,T),\|\cdot\|_{BV(0,T)}\right)$ is Banach. It possesses the following usefull properties:

- **monotonicity:** any function f in $BV(0,T)$ can be rewritten as $f=g-h$, where g and h are increasing functions, and conversely. In particular, any monotone function is in $BV(0,T)$.
- **density:** $BV(0,T)=\overline{C_c^\infty(0,T)}$, for the topology associated to the $\|\cdot\|_{BV(0,T)}$ norm.
- **compactness:** This property is a key ingredient of the proof of Theorem 5 in order to obtain convergence of the control from the convergence of a minimizing sequence of the cost function.

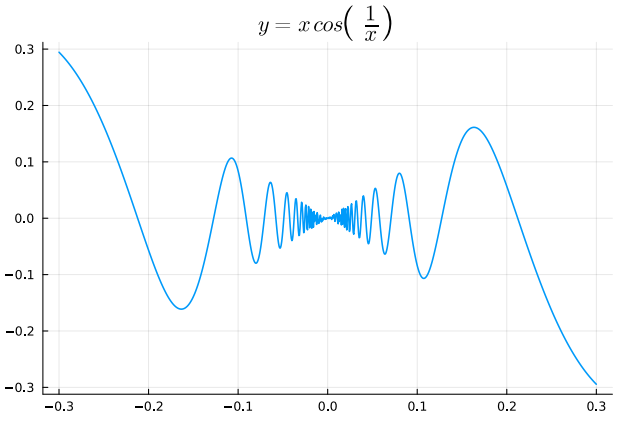
  Theorem 4. *(Helly selection theorem). for $M>0$, the function set $\{f\in BV(0,T)\mid\|f\|_{BV}\leq M\}$ is a relatively compact subset of $L^1(0,T)$, i.e. for every sequence $(f_n)_n$ in $BV(0,T)$ such that $\exists M\in\mathbb{R},\forall n, TV(f_n)\leq M$, then there exists $f\in L^1(0,T)$ such that $f_n\xrightarrow{L^1}f$, up to a subsequence.*

- **lower semi-continuity:** $BV(0,T)\ni f_n\xrightarrow{L^1_{loc}}f\Rightarrow\liminf_{n\to+\infty}TV(f_n)\geq TV(f)$. This property is also used in the proof mentionned above, to conclude that the limit control from the minimizing sequence solves the optimal control problem.

From what we've stated, BV functions are essentially an algebra of discontinuous functions that are almost everywhere differentiable. This makes this space very convenient for working with discontinuous solutions of



(a) A non-continuous function that have bounded variations: the stairs function, often encountered when modeling shockwaves.



(b) A continuous function (by continuation) which does not have bounded variations: $y(x) = x\cos(1/x)$, with instabilities in the vicinity of zero.

Figure 1.5: Examples of classical bv and non-bv functions.

PDEs, such as the shocks generated by the nonlinear scalar conservation laws we deal with in this thesis. Not only does this give us the existence of weak solutions for the Cauchy problem on network with BV initial data, but regularization properties also help to avoid the so-called "chattering" phenomenon in control theory. Indeed, having a solution in BV is morally having its gradient bounded for the L^1-norm. See chapter 2 for more on this subject.

We here give the following well known inclusions, *true on any compact interval*, to provide a global view:

$$C^1 \subset Lip \subset AC \subset BV \subset \mathcal{D} - a.e.,$$

where AC and $\mathcal{D} - a.e.$ are the sets of absolutely continuous and almost everywhere differentiable functions respectively.

Discrete framework: while the above considerations are of great help in proving most of the theoretical guarantees of our work, it is also important to highlight the behavior of the total variation on the discretizations of our problems. The discretization of a function $f : t \in [0, T] \to \mathbb{R}$ on a mesh $0 = t_0 < t_1 < \cdots < t_N = T$ leads us to consider functions of the type:

$$f^{disc}(t) := \sum_{n=0}^{N-1} f_n \mathbf{1}_{[t_n, t_{n+1}[}(t),$$

where we write $f_n := f(t_n)$, and a direct computation yields:

$$TV(f^{disc}) = \sum_{n=0}^{N-1} \mid f_{n+1} - f_n \mid .$$

Since we will often need to manipulate differentiable expressions in order to perform, for instance, a gradient

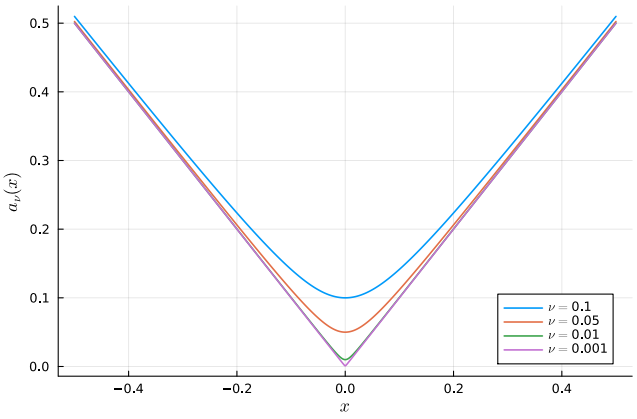


Figure 1.6: Smoothing of the absolute value with the function a_ν.

descent, we introduce for some $\nu > 0$ the *regularized total variation*, more extensively studied in subsection 2.4.1:

$$TV_\nu(f^{disc}) = \sum_{n=0}^{N-1} a_\nu(f_{n+1} - f_n).$$

It uses the following well-known smoothed absolute value, illustrated for different values of ν in Figure 1.6:

$$a_\nu(x) := \sqrt{x^2 + \nu^2}.$$

Chapter 2

Optimal scenario for road evacuation in an urban environment

The following chapter relates an article written in collaboration with Emmanuel Franck (INRIA TONUS, Strasbourg), Laurent Navoret (INRIA TONUS, Strasbourg), and Yannick Privat (IECL, Nancy).

Abstract: How to free a road from vehicle traffic as efficiently as possible and in a given time, in order to allow for example the passage of emergency vehicles? We are interested in this question which we reformulate as an optimal control problem. We consider a macroscopic road traffic model on networks, semi-discretized in space and decide to give ourselves the possibility to control the flow at junctions. Our target is to smooth the traffic along a given path within a fixed time. A parsimony constraint is imposed on the controls, in order to ensure that the optimal strategies are feasible in practice.

We perform an analysis of the resulting optimal control problem, proving the existence of an optimal control and deriving optimality conditions, which we rewrite as a single functional equation. We then use this formulation to derive a mixed algorithm interpreting it as a mix between two methods: a descent method combined with a fixed point method. We verify with numerical experiments the efficiency of this method on examples of graphs, first simple, then more complex. We highlight the efficiency of our approach by comparing it to standard methods. We propose a code implementing this approach in the Julia language, available to any user at the following link:
Keywords: Traffic network, Optimal Control, Fluid model, Hyperbolic PDE, optimization methods

AMS classification: 35Q49, 65M08, 49K15, 49K30.

2.1 Introduction

Road traffic modeling. With the concentration of populations in cities where transport flows are constantly increasing, road traffic modeling has become a central issue in urban planning. Whether it is to configure traffic lights, to adapt public transport offers or to manage in an optimal way situations of high congestion. The knowledge of the behavior of a road network is an important issue for the management of a crisis in an urban environment. It is indeed necessary to quickly predict traffic, especially traffic jams, in order to organize rescue operations. Before creating decision support tools for crisis management involving road traffic, a first step is to develop a method for monitoring road traffic.

In this work, we will focus on the evacuation of a traffic lane in finite time. It can be, for example, an axis that we wish to free in order to provide access to first aid. We approach this question in the form of an optimal control problem of a graph where each edge corresponds to a traffic lane. The main objective of this work is the determination of a prototype algorithm answering this question.

State of the art. Depending on the situation of interest, several types of approaches co-exist to model this phenomenon, depending on the scale and the desired accuracy. At the microscopic scale, trajectories (speed and position) of each vehicle can be described through Lagrangian microscopic models such as car-following models [43, 79, 62] or cellular automaton models [11, 54, 50]. Kinetic traffic models were also introduced in the 60's [26, 31, 38, 66, 71, 77, 94] and deal with vehicles on a mesoscopic scale in the form of distributions in position-velocity space.

Finally, on a macroscopic scale, the vehicle flow is then considered as a continuous medium. This is the case of the famous model introduced by Lighthill, Whitham and Richards (denoted LWR in what follows) [61, 82]. We will use it in the rest of this article, since the continuous description of the flow is particularly well adapted to the writing of a dynamic programming problem. It is notable that this first order model introduced in the 50's was later extended to Euler-type systems with viscosity, of second order [72, 95, 5, 63].

In this paper, we introduce and analyze an optimal control problem modeling an evacuation scenario for an axis belonging to a road network. A close control model has already been introduced in [37], but under restrictive assumptions about the flow regime in order to avoid congestion phenomena, which is what we want to avoid. It has been in particular highlighted that the flow may be not invertible with respect to control parameters, meaning that density at the nodes may not be reconstructed from boundary conditions. In [44], a method to manage variable speed limits combined with coordinated ramp metering within the framework of the Lighthill-Whitham-Richards (LWR) network model has been introduced. They consider a "first-discretize-then-optimize" numerical approach to solve it numerically. Note that the optimal control problem we deal with in the following has some similarities with the one proposed by the authors of this article, but differs in several respects: our control variables aim to control not just the ramps metering, but an entire axis. Furthermore, our approach to solving the underlying optimal control problem is quite different, in particular our use of optimality conditions leading to an efficient solution algorithm.

In [81], a similar control problem is discussed, which is addressed using a piecewise-linear flux framework driving to a much simplified adjoint system involving piecewise constant Jacobian matrices in time and space. This approach enables faster gradient computation, allowing them to tackle real-world simulations related to ramp-metering configuration. In our study, we opted to use the model introduced in [23] and dealt with the non-linear flux using automatic differentiation techniques to compute the gradient. This kind of approach has also been used in [8] for air traffic flow with a modified LWR-based network model.

We also mention several contributions aiming at dealing with concrete or real time cases. The paper [40] deals with this kind of control problem for simplified nonlinear (essentially without considering congestions) and linear formulations of the model, considering large scale networks. In [55], an algorithm for real-time traffic state estimation and short-term prediction is introduced.

Several other close control problems have been investigated in the past: in [46], the authors use switching controls to deal with traffic lights at an 8×4 junction in a piecewise linear flow framework. The paper [3] deals with an optimal control problem of the same kind, formulated from the continuous LWR model, solved

numerically using an heuristic random parameters search. We refer to [2, 7] for a broad view on this topic.

Regarding now controlled microscopic models, let us mention [64] where an approach based on model predictive control (MPD) is developed and [6] using a reinforcement learning algorithm. In the book [65], the last chapter is devoted to the modeling of various optimization problems related to road traffic. There has also been some research into the optimal switching of the traffic lights to maximize the traffic flow using a mixed-integer model [46]. Other works have looked at traffic regulation by comparing and discussing the use of lights and circles [17].

Concerning now the control of mesoscopic and macroscopic models, there are software contributions seeking in particular to bring real time answers and guidance tools in case of accident. Let us also mention [48, 33] dealing in part with traffic prediction and control problems.

This article uses a controlled model close to the one studied in [37] which is devoted to the control of the macroscopic LWR model on a directed graph using the framework introduced in [23]. However, we are interested in taking into account some properties of the model that were not addressed in [37], such as congestion phenomena.

Hence, we will not focus on individual paths since we are interested in the overall dynamics with a focus on congestion. Particular attention is paid to the modeling of junctions as they introduce a nonlinear coupling between roads.

Main objectives. Our goal is to design a numerical method to control the flow of vehicles in all fluid regimes, including saturated regimes/congestions, using dynamical barriers at each road in the graph.

We thus propose a road traffic model in the form of a controlled graph at each junction. This models an urban area structured by roads. For practical reasons, even if we wish to take into account possible congestion/shocks in the system, we position ourselves in a differentiable framework by adding weak diffusion terms in the system, through a semi-discretization in space. Hence, the resulting problem consists in minimizing the vehicle density at the final time on a given route under ODE constraint, obtained by finite volume semi-discretization of the LWR system. We then enrich this problem with a constraint on the number of simultaneously active roadblocks in order to take into account staffing issues.

Plan of this article. While the macroscopic model describing the flow is very standard (Lighthill-Whitham-Richards, 1955) the problem of flow distribution at junctions, inspired by more recent works [41],[23], realizes a nonlinear coupling between the different edges. The distribution of vehicles at junctions is thus modeled by an optimal allocation process that depends on the maximum possible flows at the junctions, through a linear programming (LP) problem targeting the maximization of incoming flows. In order to influence the traffic flow, we introduce control functions that are defined at each road entrance and acts as barriers by weighting the capacity of a road leaving a junction to admit new vehicles.

After investigating the existence of optimal control, we then derive necessary optimality conditions that we reformulate in an exploitable way and introduce an optimization algorithm based on a hybrid combination of a variable step gradient method and a well-chosen fixed point method translating the optimality conditions. We then illustrate the efficiency of the introduced method using examples of various graphs modeling, in particular, a traffic circle or a main road surrounded by satellite roads.

Our contribution can be summarized in the form of a computational code allowing to process complex graphs,

but nevertheless of rather small dimension. This code, written in the Julia language, is available and usable at the following link:

https://github.com/mickaelbestard/TRoN.jl

Notations. Throughout this article, we will use the following notations:

- For $x \in \mathbb{R}^n$, x_+ will denote the positive part of x, namely $x_+ = \max\{x, 0_{\mathbb{R}^n}\}$, the max being understood component by component;
- $\mathrm{BV}(0,T)$ denotes the space of all functions of bounded variation on $(0,T)$;
- $W^{1,\infty}(0,T)$ denotes the space of all functions f in $L^\infty(0,T)$ whose gradient in the sense of distributions also belongs to $L^\infty(0,T)$;
- $\|\cdot\|_{\mathbb{R}^d}$ (resp. $\langle\cdot,\cdot\rangle_{\mathbb{R}^d}$) stands for the standard Euclidean norm (resp. inner product) in $\mathbb{R}^d$;
- Let $F : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}^m$ and $(x_0, y_0) \in \mathbb{R}^p \times \mathbb{R}^q$. We will denote by $\partial_x F(x_0, y_0)$, the Jacobian matrix of the mapping $x \mapsto F(x, y_0)$. When no ambiguity is possible, we will simply denote it by $\partial_x F$;
- N_r: number of roads in the model;
- N_c number of mesh cells per road for the first-order Finite Volumes scheme;
- N_T: number of time discretization steps in the numerical schemes;

2.2 A controlled model of trafic flow

In this section, we first present the traffic flow model on a road network and its semi-discretization. Then we introduce the precise control model: the control at junctions will translates regulation action using traffic signs or traffic lights.

2.2.1 Traffic dynamics on network without control

The road network is a directed graph of N_r roads, with $N_r \in \mathbb{N}^*$, where each edge corresponds to a road and each vertex to a junction. The roads will all be denoted as real intervals $[a_i, b_i]$ for some $i \in [\![1, N_r]\!]$. Of course, this writing does not determine the topology of the graph. It is necessary to define, for each junction, the indices of the incoming and outgoing roads in order to characterize the directed graph completely.

We are interested in the time evolution of the densities on each road. On the i-th road, the evolution of the density $\rho_i(t,x)$ is provided by the so-called LWR model

$$\begin{cases} \partial_t \rho_i(t,x) + \partial_x f_i(\rho_i(t,x)) = 0, & (t,x) \in (0,T) \times [a_i, b_i], \\ \rho_i(0,x) = \rho_i^0(x), & x \in [a_i, b_i], \\ f_i(\rho_i(t,a_i)) = \gamma_i^L(t), & t \in (0,T), \\ f_i(\rho_i(t,b_i)) = \gamma_i^R(t), & t \in (0,T), \end{cases} \tag{LWR}$$

where the flux $f_i(\rho)$ is given by

$$f_i(\rho) = \rho\, v_i^{\max}\left(1 - \frac{\rho}{\rho_i^{\max}}\right), \tag{2.1}$$

where $v_i^{\max}$ and $\rho_i^{\max}$ denotes respectively the maximal velocity and density allowed on the i-th road. We will consider without loss of generality that $\rho_i^{\max} = v_i^{\max} = 1$. The initial density is denoted $\rho_i^0(x)$ and $\boldsymbol{\gamma}_i^L$ and $\boldsymbol{\gamma}_i^R$ are functions allowing to prescribe the flux at the left and right boundaries of the interval.

The fluxes $\boldsymbol{\gamma}$ at junctions are determined using the model introduced in [23]. Considering a junction J, the sets of indices corresponding to the in-going and out-going roads are denoted J_{in} and J_{out}. We assume that there is a statistical behavior matrix A_J defined by

$$A_J := (\alpha_{ji})_{(j,i)\in J_{\text{out}}\times J_{\text{in}}}, \quad 0 < \alpha_{ji} < 1, \quad \sum_{j\in J_{\text{out}}} \alpha_{ji} = 1,$$

with α_{ji} the proportion of vehicles going to the j-th outgoing road among those coming from the i-th ingoing road. Outgoing fluxes have to satisfy the relation

$$(\gamma_j^L)_{j\in J_{\text{out}}} = A_J(\gamma_i^R)_{i\in J_{\text{in}}}, \tag{2.2}$$

and we have the balance between in-going and out-going fluxes:

$$\sum_{i\in J_{\text{in}}} \gamma_i^R = \sum_{j\in J_{\text{out}}} \gamma_j^L.$$

Let us note that we have natural constraints on the fluxes. Indeed the LWR flux function f_i reaches its maximum at $\rho_i^{\max}/2$. Therefore, ingoing fluxes γ_i^R have to be smaller than the following upper bound:

$$\gamma_i^{R,\max} = f_i(\min\{\rho_i(t,b_i), \rho_i^{\max}/2\}),$$

which takes into account reduced demands when the density is lower than $\rho_i^{\max}/2$ in the ingoing road. Similarly outgoing fluxes γ_j^L have to be smaller than the upper bound:

$$\gamma_j^{L,\max} = f_j(\max\{\rho_j(t,a_j), \rho_j^{\max}/2\}),$$

which takes into account reduced capacities when the density is larger than $\rho_i^{\max}/2$ in the outgoing road. We refer to [23] for details. Consequently, the fluxes belong to following set:

$$\Omega_J = \left\{ \left((\gamma_i^R)_{j\in J_{\text{in}}}, (\gamma_j^L)_{i\in J_{\text{out}}}\right) \in \prod_{i\in J_{\text{in}}} \left[0, \gamma_i^{R,\max}\right] \prod_{j\in J_{\text{out}}} \left[0, \gamma_j^{L,\max}\right] \quad \text{with} \quad (\gamma_j^L)_{j\in J_{\text{out}}} = A_J(\gamma_i^R)_{i\in J_{\text{in}}} \right\}$$

Then we assume that drivers succeed in maximizing the total flow. Consequently the fluxes are solution to the following Linear Programming (LP) problem:

$$\max_{\left((\gamma_i^R)_{j\in J_{\text{in}}}, (\gamma_j^L)_{j\in J_{\text{out}}}\right)\in\Omega_J} \sum_{i\in J_{\text{in}}} \gamma_i^R, \tag{2.3}$$

This definition is valid at any junction of the network. Note that this problem does not have necessarily a unique solution in the case where there are more incoming roads than outgoing roads. In that case, a priority modeling assumption should be added to select one solution. We refer to [14] for more details. In the sequel, we suppose that this linear programming problem have a unique solution solution and we formally write:

$$\boldsymbol{\gamma}(t) = \phi^{LP}(\boldsymbol{\rho}(t)),$$

for the simultaneous resolution of the linear programming problems at all the junctions of the network. Note that the dependency on $\boldsymbol{\rho}(t)$ results from the definition of the upper bounds involved in the definition of the sets Ω_J.

In the following, we will only consider networks with at most two ingoing roads and two outgoing roads, i.e. either 1×1, 1×2, 2×1 or 2×2 junctions. This gives existence and stability of the LWR Cauchy problem [23], and has the advantage that the function ϕ^{LP} can be explicitly provided [98] while it already enables to model a huge variety of road networks.

2.2.2 Control at junctions

We introduce the vector of controls $\mathbf{u} = (u_i(\cdot))_{1 \leq i \leq N_r} \in L^\infty([0,T], \mathbb{R}^{N_r})$ at every road entrance. Note that we implicitly assume that every road junction is controlled, but our model allows without any difficulty to neutralize some controls in order to model the fact that only some junctions are controlled.

We interpret each control u_j as a rate, assuming that at each time $t \in (0,T)$, the maximum flow out a junction and going into road j is weighted by a coefficient $u_j(t) \in [0,1]$. This allow in particular to keep valid all well-posedness considerations mentioned in [23]. At junction J, we thus define the following polytope of constraints

$$\Omega_J(\mathbf{u}) = \Big\{ \big((\gamma_i^R)_{j \in J_{\text{in}}}, (\gamma_j^L)_{j \in J_{\text{out}}}\big) \in \prod_{i \in J_{\text{in}}} \left[0, \gamma_i^{R,\max}\right] \prod_{j \in J_{\text{out}}} \left[0, (1-u_j)\gamma_j^{L,\max}\right] \\ \text{with} \quad (\gamma_j^L)_{j \in J_{\text{out}}} = A_J (\gamma_i^R)_{i \in J_{\text{in}}} \Big\}$$

This initial model is not completely satisfactory since a full control on one outgoing road would result on zero outgoing fluxes for all the ingoing roads. Indeed relation (2.2) implies that ingoing fluxes are linear combination of outgoing fluxes with positive weights. This is definitely not the desired behavior as we would expect that the trafic flow would be deviated to the uncontrolled outgoing roads. To solve this problem, we choose to make the statistical behavior matrix A_J also dependent on the control $\mathbf{u}$. More precisely, we ask that, as soon as a road entry is fully controlled and the other are not controlled, the proportion of vehicles entering the roas is set to 0. To illustrate this point, let us describe the control of 1×2 junction, i.e. with one incoming and 2 outgoing roads. Then the traffic distribution matrix A writes:

$$A(\mathbf{u}) = \begin{pmatrix} \alpha(\mathbf{u}) \\ 1 - \alpha(\mathbf{u}) \end{pmatrix},$$

where $\alpha(\mathbf{u})$ denotes the proportion of vehicles going in the first outgoing road. Then, denoting $\mathbf{u} = (u_1, u_2)$ the controls of the two outgoing roads, the function $\alpha(u_1, u_2)$ has to be chosen such that:

(a) if the first outgoing road is fully controlled and the second is not controlled, then $\alpha(\mathbf{u})$ has to vanish, resulting

in the condition: $\alpha(1,0)=0$, and then the entire vehicle flow goes in the second outgoing road.

(b) inversely, if the second outgoing road is fully controlled and the first one is not controlled, then $1-\alpha(\mathbf{u})$ has to vanish, resulting in the condition: $1-\alpha(0,1)=0$, and then the entire vehicle flow goes in the first outgoing road.

We also ask for the following additional property:

(c) the parameters are not modified if the outgoing roads are equally controlled, which results in the condition:

$$\forall u\in[0,1],\quad \alpha(u,u)=\bar{\alpha},$$

where $\bar{\alpha}$ is a given distribution coefficient.

In Appendix 2.6, we provide an explicit construction of such a function. We also treat the case of 2×2 junctions. Note that junctions with only one outgoing roads (1×1 and 2×1) do not require such modification.

Like in the previous section, the simultaneous resolution of the Linear Porgramming problems at each junctions junction is now denoted:

$$\boldsymbol{\gamma}=\phi^{LP}(\boldsymbol{\rho},\mathbf{u})$$

The explicit expressions of ϕ^{LP} in the set of cases we deal with are provided in Appendix 2.6. Beyond the explicit expression of $\phi^{LP}(\boldsymbol{\rho},\mathbf{u})$, what matters is that ϕ^{LP} is a Lipschitz function with respect to $(\boldsymbol{\rho},\mathbf{u})$. In the following, in order to use tools of differentiable optimization, we will consider a C^1 approximation of this function. This issue is commented at the end of Appendix 2.6. According to these comments, we will assume from now on:

The function ϕ^{LP} is Lipschitz, and C^1 with respect to its second variable $\mathbf{u}$. ($\mathrm{H}_{\phi^{LP}}$)

2.2.3 Semi-discretized model

For algorithmic efficiency reasons, we prefer to be able to define the sensitivity of the different data of the problem with respect to the control. Since the model (LWR) is known to generate possible irregularities in the form of shocks, we have decided to introduce regularity through a semi-discretization of the model in space. The relevance of this choice will be discussed in the concluding section.

Hence, we discretize the model (LWR) with a first-order Finite Volume (FV) scheme. We consider $N_c\in\mathbb{N}^*$ mesh cells per road: the discretization points on the i-th roads are denoted $a_i=x_{i,1/2}<x_{i,3/2}<\ldots<x_{i,N_c-1/2}=b_i$ and the space steps $\triangle x_{i,j}=x_{i,j+1/2}-x_{i,j-1/2}$. Then the discrete densities and the boundary fluxes are denoted:

$$\boldsymbol{\rho}(t)=\big(\rho_{i,j}(t)\big)_{\substack{1\le i\le N_r\\1\le j\le N_c}}=\left(\frac{1}{\triangle x_{i,j}}\int_{x_{i,j-\frac{1}{2}}}^{x_{i,j+\frac{1}{2}}}\rho(t,x)\,\mathrm{d}x\right)_{\substack{1\le i\le N_r\\1\le j\le N_c}},$$

$$\boldsymbol{\gamma}(t)=\big(\gamma_1^L(t),\gamma_1^R(t),\ldots,\gamma_{N_r}^L(t),\gamma_{N_r}^R(t)\big)\,.$$

With a slight abuse of notation, we have written similarly the discrete variable as the continuous one, since we will essentially work on the semi-discretized model. Then the semi-discretized dynamics is given by the

differential equation

$$\begin{cases} \dfrac{\mathrm{d}\boldsymbol{\rho}}{\mathrm{d}t}(t) = f^{FV}\left(\boldsymbol{\rho}(t), \boldsymbol{\gamma}(t)\right), & t \in (0,T) \\ \boldsymbol{\gamma}(t) = \phi^{LP}\left(\boldsymbol{\rho}(t), \mathbf{u}(t)\right)), & t \in (0,T) \\ \boldsymbol{\rho}(0) = \boldsymbol{\rho}_0, \end{cases} \tag{LWR-sd}$$

where the finite volume flow at the j-th mesh cell of the i-th road reads

$$f^{FV}(\boldsymbol{\rho}, \boldsymbol{\gamma})_{ij} = \begin{cases} \frac{-1}{\triangle x_{ij}} \left(\mathcal{F}_i(\rho_{i,j}, \rho_{i,j+1}) - \gamma_i^L\right), & \text{if } j = 1, \\ \frac{-1}{\triangle x_{ij}} \left(\mathcal{F}_i(\rho_{i,j}, \rho_{i,j+1}) - \mathcal{F}_i(\rho_{i,j-1}, \rho_{i,j})\right), & \text{if } 2 \leq j \leq N_c - 1, \\ \frac{-1}{\triangle x_{ij}} \left(\gamma_i^R - \mathcal{F}_i(\rho_{i,j-1}, \rho_{i,j})\right), & \text{if } j = N_c, \end{cases} \tag{2.4}$$

with $\mathcal{F}_i(u,v)$ the so-called local Lax-Friedrich numerical flux given by

$$\mathcal{F}_i(u,v) = \frac{f_i(u) + f_i(v)}{2} - \max\left\{|f_i'(u)|, |f_i'(v)|\right\}\frac{(v-u)}{2}.$$

We refer to [59] for an introduction to Finite Volume approximations.

2.2.4 Conclusion: an optimal control problem

We are interested in an approximate controllability problem which consists in emptying a given route *as much as possible* for a fixed end time $T > 0$. Let us introduce $\chi_{\text{path}} \subset [\![1, N_r]\!]$, a set of indices corresponding to the route we wish to empty in a time T.

We would like to minimize a functional with respect to the control $\mathbf{u}$, representing the sum of all the densities on this path, in other words

$$C_T(\mathbf{u}) = \sum_{i \in \chi_{\text{path}}} \sum_{j=1}^{N_c} \rho_{i,j}(T; \mathbf{u}) = \mathbf{c} \cdot \boldsymbol{\rho}(T; \mathbf{u}),$$

where $\boldsymbol{\rho}(t; \mathbf{u})$ denotes the solution to the controlled system (LWR-sd) and $\mathbf{c} = (c_i)_{1 \leq i \leq N_r}$ the vector defined by

$$c_i = 1 \text{ if } i \in \chi_{\text{path}} \quad \text{and} \quad c_i = 0 \text{ else.}$$

Of course, it is necessary to introduce a certain number of constraints on the sought controls, in accordance with the model under consideration, and to model that the obtained control is feasible in practice. We are thus driven to consider the following constraints:

(i) $0 \leq u_i(\cdot) \leq 1$, meaning that at each time, the control is a vehicle acceptance rate on a road;

(ii) $\sum_{i=1}^{N_r} u_i(\cdot) \leq N_{\max}$: we impose at each time a maximum number of active controls in order to take into account the staff required for roadblocking.

(iii) Regularity: we will assume that $\mathbf{u}$ is of bounded variation, and write $\mathbf{u} \in \mathrm{BV}([0,T]; \mathbb{R}^{N_r})$. This constraint

involves the *total variation*[1] of the control and models that the roadblock is supposed not to "blink" over time. From a control point of view, we aim at avoiding the so-called *chattering phenomenon*.

The first constraint above will be included in the set of admissible controls. Concerning the other two constraints, we have chosen to include them as penalty/regularization terms in the functional. Of course, other choices would be quite possible and relevant.

Let $N_{\max} \in \mathbb{N}^*$ be an integer standing for the maximal number of active controls for this problem and $\theta = (\theta_S, \theta_B) \in \mathbb{R}^2$ be two non-negative parameters. According to all the considerations above, the optimal control problem we will investigate reads:

$$\boxed{\inf_{\mathbf{u}\in\mathcal{U}_{ad}} \mathcal{J}(\mathbf{u}),} \tag{$\mathcal{P}_\theta$}$$

where the admissible set of controls is defined by

$$\mathcal{U}_{ad} = L^\infty([0,T],[0,1]^{N_r}), \tag{2.5}$$

and the regularized cost functional $\mathcal{J}$ writes

$$\mathcal{J}(\mathbf{u}) = C_T(\mathbf{u}) + \theta_S S(\mathbf{u}) + \theta_B B(\mathbf{u}), \tag{2.6}$$

where $S(\mathbf{u})$ denotes the regularizing term modeling the limitation on the number of active controls and $B(\mathbf{u})$ is the total variation of $\mathbf{u}$ in time, namely

$$S(\mathbf{u}) = \frac{1}{2}\int_0^T \left(\sum_{i=1}^{N_r} u_i(t) - N_{max}\right)_+^2 dt \quad \text{and} \quad B(\mathbf{u}) = \sum_{i=1}^{N_r} \mathrm{TV}(u_i). \tag{2.7}$$

2.3 Analysis of the optimal control problem ($\mathcal{P}_\theta$)

In this section, we will investigate the well-posedness and derive the optimality conditions of Problem ($\mathcal{P}_\theta$). These conditions will form the basis of the algorithms used in the rest of this study.

2.3.1 Well-posedness of Problem ($\mathcal{P}_\theta$)

We follow the so-called direct method of calculus of variations. The key point of the following result is the establishment of uniform estimates of $\boldsymbol{\rho}$ with respect to the control variable $\mathbf{u}$, in the $W^{1,\infty}(0,T)$ norm.

Theorem 5. *Let us assume that θ_B is positive. Then, Problem ($\mathcal{P}_\theta$) has a solution.*

Proof. Let $(\mathbf{u}_n)_{n\in\mathbb{N}}$ be a minimizing sequence for Problem ($\mathcal{P}_\theta$). Observe first that, by minimality, the sequence $(\mathcal{J}(\mathbf{u}_n))_{n\in\mathbb{N}}$ is bounded, and therefore, so is the total variation $(B(\mathbf{u}_n))_{n\in\mathbb{N}}$. We infer that, up to a subsequence, $(\mathbf{u}_n)_{n\in\mathbb{N}}$ converges in $L^1(0,T)$ and almost everywhere toward an element $\mathbf{u}_* \in \mathrm{BV}(0,T)$. Since the class $\mathcal{U}_{ad}$ is closed for this convergence, we moreover get that $\mathbf{u}_* \in L^\infty([0,T],[0,1]^{N_r})$.

[1] Given a function f belonging to $L^1([0,T])$, the total variation of f in $[0,T]$ is defined as

$$\mathrm{TV}(f) = \sup\left\{\int_0^T f(t)\phi'(t)\,dt,\ \phi \in C_c^1([0,T]),\ \|\phi\|_{L^\infty([0,T])} \le 1\right\}.$$

For the sake of readability, we will denote similarly a sequence and any subsequence with a slight abuse of notation.

In what follows, it is convenient to introduce the function $g : \mathbb{R}^{N_c \times N_r} \times [0,1]^{N_r}$ defined by $g(\boldsymbol{\rho}, \mathbf{u}) = f^{FV}(\boldsymbol{\rho}, \phi^{LP}(\boldsymbol{\rho}, \mathbf{u}))$, so that $\boldsymbol{\rho}$ solves the ODE system $\boldsymbol{\rho}' = g(\boldsymbol{\rho}, \mathbf{u})$. Let us set $\boldsymbol{\rho}_n := \boldsymbol{\rho}(\cdot\,; \mathbf{u}_n)$.

Step 1: the function g is continuous. Indeed, note that the function ϕ^{LP} is continuous (and even Lipschitz), according to ($H_{\phi^{LP}}$). We also refer to the comments of modeling issues at the end of Section 2.2.2. According to (2.4), f^{FV} has therefore the same regularity as the numerical flow $\boldsymbol{\rho} \mapsto \mathcal{F}(\boldsymbol{\rho})$, defined by

$$\mathcal{F}_i^{j,j+1}(\boldsymbol{\rho}) = \frac{f_i^j + f_i^{j+1}}{2} - \frac{\max\{|df_i^j|, |df_i^{j+1}|\}}{2}\left(\rho_i^{j+1} - \rho_i^j\right),$$

with $f_i^j = \rho_i^j(1-\rho_i^j)$ and $df_i^j = 1 - 2\rho_i^j$, which is obviously continuous.

Step 2: the sequence $(\boldsymbol{\rho}_n)_{n\in\mathbb{N}}$ is uniformly bounded in $W^{1,\infty}(0,T)$. For $\mathbf{u} \in \mathcal{U}_{ad}$ given, let us introduce the function $\mathcal{L} : t \mapsto \frac{1}{2}\|\rho(t)\|^2_{\mathbb{R}^{N_c\times N_r}}$. Up to standard renormalizations, we will suppose in this proof that $v_i^{\max} = \rho_i^{\max} = 1$ and that Δx_i does not depend on the index i. According to the chain rule, $\mathcal{L}$ is differentiable, and its derivative reads[2]

$$\begin{aligned}
\mathcal{L}'(t) &= \langle \boldsymbol{\rho}'(t), \boldsymbol{\rho}(t)\rangle_{\mathbb{R}^{N_c\times N_r}} = \langle g(\boldsymbol{\rho}(t), \mathbf{u}(t)), \boldsymbol{\rho}(t)\rangle_{\mathbb{R}^{N_c\times N_r}} \\
&= -\frac{1}{\Delta x}\sum_{i=1}^{N_r}\sum_{j=1}^{N_c}\left(\mathcal{F}_i^{j,j+1}(\boldsymbol{\rho}) - \mathcal{F}_i^{j-1,j}(\boldsymbol{\rho})\right)\rho_i^j \\
&= -\frac{1}{\Delta x}\sum_{i=1}^{N_r}\left(\gamma_i^R\rho_i^{N_c} + \sum_{j=1}^{N_c-1}\mathcal{F}_i^{j,j+1}(\boldsymbol{\rho})\rho_i^j - \gamma_i^L\rho_i^1 - \sum_{j=2}^{N_c}\mathcal{F}_i^{j-1,j}(\boldsymbol{\rho})\rho_i^j\right) \\
&= \frac{1}{\Delta x}\sum_{i=1}^{N_r}\left(\gamma_i^L\rho_i^1 - \gamma_i^R\rho_i^{N_c} + \sum_{j=1}^{N_c-1}\mathcal{F}_i^{j,j+1}(\boldsymbol{\rho})\left(\rho_i^{j+1} - \rho_i^j\right)\right).
\end{aligned}$$

For a given $i \in [\![1, N_r]\!]$, and $j \in [\![1, N_c - 1]\!]$, we have

$$\begin{aligned}
\mathcal{F}_i^{j,j+1}(\boldsymbol{\rho})\left(\rho_i^{j+1} - \rho_i^j\right) &= \left(\frac{f_i^j + f_i^{j+1}}{2} - \frac{\max\{|df_i^j|, |df_i^{j+1}|\}}{2}\left(\rho_i^{j+1} - \rho_i^j\right)\right)\left(\rho_i^{j+1} - \rho_i^j\right) \\
&\le \left(f_i(\sigma_i) + (\rho_i^{j+1} - \rho_i^j)\right)\left(\rho_i^{j+1} - \rho_i^j\right),
\end{aligned}$$

since $f_i^j \le f_i(\sigma_i) = 1/4$ according to the explicit expression (2.1) of f_i, and $|df_i^j| \le v_i^{\max} = 1$. Since the local Lax-Friedrich numerical scheme is monotone, we infer that ρ_i^j is lower than $\rho_{\max} = 1$. We then have

$$\begin{aligned}
\mathcal{F}_i^{j,j+1}(\boldsymbol{\rho})\left(\rho_i^{j+1} - \rho_i^j\right) &\le 2f_i(\sigma_i) + \left(\rho_i^{j+1} - \rho_i^j\right)^2 = \frac{1}{2} + \left(\rho_i^{j+1} - \rho_i^j\right)^2 \\
&\le \frac{1}{2} + (\rho_i^j)^2 + (\rho_i^{j+1})^2 - 2\rho_i^j\rho_i^{j+1} \\
&\le \frac{1}{2} + (\rho_i^j)^2 + (\rho_i^{j+1})^2,
\end{aligned}$$

[2] we drop the time dependancy notation for readability.

by positivity of the ρ_i^j. Using the majoration $\gamma_i^L \rho_i^1 - \gamma_i^R \rho_i^{N_c} \leq f_i(\sigma_i) \times 2 = 1/2$, all the calculations above yield

$$\begin{aligned}\mathcal{L}'(t) &\leq \frac{1}{\Delta x} \sum_{i=1}^{N_r} \left(\frac{1}{2} + \sum_{j=1}^{N_c-1} \left(\frac{1}{2} + (\rho_i^j)^2 + (\rho_i^{j+1})^2 \right) \right) \\ &\leq \frac{1}{\Delta x} \left(\frac{N_r N_c}{2} + ||\boldsymbol{\rho}(t)||^2 \right).\end{aligned}$$

We infer the existence of two positive numbers $\bar{\alpha}$, $\bar{\beta}$ that do not depend on $\mathbf{u}$, such that $\mathcal{L}'(t) \leq \bar{\alpha}\mathcal{L}(t) + \bar{\beta}$ for a.e. $t \in (0,T)$. By using a Grönwall-type inequality, we get

$$\mathcal{L}(t) \leq \mathcal{L}(0)e^{\bar{\alpha}t} + \int_0^t \beta e^{\bar{\alpha}(t-s)}\, ds = \left(\mathcal{L}(0) + \frac{\bar{\beta}}{\bar{\alpha}}(1 - e^{-\bar{\alpha}t}) \right) e^{\bar{\alpha}t}, \tag{2.8}$$

Step 3: conclusion. According to (2.8), the sequence $(\boldsymbol{\rho}_n)_{n\in\mathbb{N}}$ is bounded in $L^\infty(0,T)$. Since $\boldsymbol{\rho}_n' = g(\boldsymbol{\rho}_n, \mathbf{u}_n)$ a.e. in $(0,T)$ and g is continuous, it follows that $(\boldsymbol{\rho}_n)_{n\in\mathbb{N}}$ is uniformly bounded in $W^{1,\infty}(0,T)$. Therefore, by using the Arzela-Ascoli theorem, this sequence converges up to a subsequence in $C^0([0,T])$ toward an element $\boldsymbol{\rho}_* \in W^{1,\infty}(0,T)$.

Let us recast System (LWR-sd) as a fixed point equation, as

$$\text{for every } t \in (0,T), \quad \boldsymbol{\rho}_n(t) - \boldsymbol{\rho}_0 = \int_0^t g(\boldsymbol{\rho}_n(s), \mathbf{u}_n(s))\, \mathrm{d}s,$$

we obtain by letting n go to $+\infty$,

$$\boldsymbol{\rho}_*(t) - \boldsymbol{\rho}_0 = \int_0^t g(\boldsymbol{\rho}_*(s), \mathbf{u}_*(s))\, \mathrm{d}s,$$

meaning that $\boldsymbol{\rho}^*$ solves System (LWR-sd) with $\mathbf{u}^*$ as control. Finally, according to the aforementioned convergences and since the functionals S and B are convex, it is standard that one has

$$\lim_{n\to+\infty} C_T(\mathbf{u}_n) = C_T(\mathbf{u}^*), \quad \liminf_{n\to+\infty} S(\mathbf{u}_n) \geq S(\mathbf{u}^*), \quad \liminf_{n\to+\infty} B(\mathbf{u}_n) \geq B(\mathbf{u}^*).$$

We get that $\mathcal{J}(\mathbf{u}^*) \leq \liminf_{n\to+\infty} \mathcal{J}(\mathbf{u}_n)$, and we infer that $\mathbf{u}^*$ solves Problem ($\mathcal{P}_\theta$). ■

2.3.2 Optimality conditions

We have established the existence of an optimal control in Theorem 5. In order to derive a numerical solution algorithm, we will now state the necessary optimality conditions on which the algorithm we will build is based. One of the difficulties is that the functional we use involves non-differentiable quantities. We will therefore first use the notion of subdifferential to write the optimality conditions. In Section 2.4 dedicated to numerical experiments, we will explain how we approximate these quantities.

Let us first compute the differential of the functional C_T. To this aim, we introduce the tangent cone to the set $\mathcal{U}_{ad}$.

Definition 4. *Let $\mathbf{u} \in \mathcal{U}_{ad}$. A function $\mathbf{h}$ in $L^\infty(0,T)$ is said to be an **admissible perturbation** of $\mathbf{u}$ in $\mathcal{U}_{ad}$ if, for every sequence of positive real numbers $(\varepsilon_n)_{n\in\mathbb{N}}$ decreasing to 0, there exists a sequence of functions $\mathbf{h}^n$*

converging to $\mathbf{h}$ *for the weak-star topology of* $L^\infty(0,T)$ *as* $n \to +\infty$*, and such that* $\mathbf{u} + \varepsilon_n \mathbf{h}^n \in \mathcal{U}_{ad}$ *for every* $n \in \mathbb{N}$.

Proposition 6. *Let* $\mathbf{u} \in \mathcal{U}_{ad}$ *and* $(\boldsymbol{\rho}, \boldsymbol{\gamma})$ *the associated solution to* (LWR-sd)*. We introduce the two matrices* M *and* N *defined from the Jacobian matrices of* f^{FV} *and* ϕ^{LP} *as*

$$M(\boldsymbol{\rho}, \boldsymbol{\gamma}, \mathbf{u}) = (\partial_{\boldsymbol{\rho}} f^{FV})(\boldsymbol{\rho}, \boldsymbol{\gamma}) + (\partial_{\boldsymbol{\gamma}} f^{FV})(\boldsymbol{\rho}, \boldsymbol{\gamma})\, (\partial_{\boldsymbol{\rho}} \phi^{LP})(\boldsymbol{\rho}, \mathbf{u}), \tag{2.9}$$

$$N(\boldsymbol{\rho}, \boldsymbol{\gamma}, \mathbf{u}) = (\partial_{\boldsymbol{\gamma}} f^{FV})(\boldsymbol{\rho}, \boldsymbol{\gamma})(\partial_{\mathbf{u}} \phi^{LP})(\boldsymbol{\rho}, \mathbf{u}), \tag{2.10}$$

where we use the notational conventions introduced in Section 2.1.

The functional C_T *is differentiable in the sense of Gâteaux and its differential reads*

$$\mathrm{d}C_T(\mathbf{u})\mathbf{h} = \int_0^T (N(\boldsymbol{\rho}, \boldsymbol{\gamma}, \mathbf{u})^\top \mathbf{p}) \cdot \mathbf{h}\, \mathrm{d}t, \tag{2.11}$$

for every admissible perturbation $\mathbf{h}$*, where* $\mathbf{p}$ *is the so-called adjoint state, defined as the unique solution to the Cauchy system*

$$\begin{cases} \mathbf{p}' + M(\boldsymbol{\rho}, \boldsymbol{\gamma}, \mathbf{u})^\top \mathbf{p} = 0 \quad \textit{in } (0,T), \\ \mathbf{p}(T) = \mathbf{c}. \end{cases} \tag{2.12}$$

Remark 2. *In Proposition 6 above, the matrices* M *and* N *express respectively the way by which* $\boldsymbol{\rho}$ *interacts with* $\boldsymbol{\gamma}$ *and* $\boldsymbol{\gamma}$ *interacts with* $\mathbf{u}$*.*

Proof of Proposition 6. Let $\mathbf{u} \in \mathcal{U}_{ad}$. The Gâteaux differentiability of C_T, $\mathcal{U}_{ad} \ni \mathbf{u} \mapsto \boldsymbol{\rho}$ and $\mathcal{U}_{ad} \ni \mathbf{u} \mapsto \boldsymbol{\gamma}$ is standard, and follows for instance directly of the proof of the Pontryagin Maximum Principle (PMP, see e.g. [58]). Although the expression of the differential of C_T could also be obtained by using the PMP, we provide a short proof hereafter to make this article self-contained.

Let $\mathbf{h} \in L^\infty(0,T,\mathcal{U})$ be an admissible perturbation of $\mathbf{u}$ in $\mathcal{U}_{ad}$. One has

$$\mathrm{d}C_T(\mathbf{u})\, \mathbf{h} = \mathbf{c} \cdot \dot{\boldsymbol{\rho}}(T), \tag{2.13}$$

where $\dot{\boldsymbol{\rho}} = \mathrm{d}(\mathbf{u} \mapsto \boldsymbol{\rho})\mathbf{h}$ solves the system

$$\begin{cases} \dot{\boldsymbol{\rho}}' = (\partial_{\boldsymbol{\rho}} f^{FV})(\boldsymbol{\rho}, \boldsymbol{\gamma})\dot{\boldsymbol{\rho}} + (\partial_{\boldsymbol{\gamma}} f^{FV})(\boldsymbol{\rho}, \boldsymbol{\gamma})\dot{\boldsymbol{\gamma}}, \\ \dot{\boldsymbol{\gamma}} = (\partial_{\boldsymbol{\rho}} \phi^{LP})(\boldsymbol{\rho}, \mathbf{u})\dot{\boldsymbol{\rho}} + (\partial_{\mathbf{u}} \phi^{LP})(\boldsymbol{\rho}, \mathbf{u})\mathbf{h}, \\ \dot{\boldsymbol{\rho}}(0) = 0. \end{cases}$$

We infer that $\dot{\boldsymbol{\rho}}$ solves

$$\begin{cases} \dot{\boldsymbol{\rho}}' = M(\boldsymbol{\rho}, \boldsymbol{\gamma}, \mathbf{u})\dot{\boldsymbol{\rho}} + N(\boldsymbol{\rho}, \boldsymbol{\gamma}, \mathbf{u})\mathbf{h}, \\ \dot{\boldsymbol{\rho}}(0) = 0, \end{cases} \tag{2.14}$$

with M and N as defined in (2.9). Let us multiply the main equation of (2.12) by $\dot{\boldsymbol{\rho}}$ in the sense of the inner product, and integrate over $(0,T)$. We obtain:

$$\int_0^T \dot{\boldsymbol{\rho}} \cdot \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t}\, \mathrm{d}t + \int_0^T \dot{\boldsymbol{\rho}} \cdot (M(\boldsymbol{\rho}, \boldsymbol{\gamma}, \mathbf{u})^\top \mathbf{p})\, \mathrm{d}t = 0.$$

Similarly, let us multiply the main equation of (2.14) by $\mathbf{p}$ in the sense of the inner product, and integrate over $(0,T)$. We obtain:

$$\int_0^T \mathbf{p}\cdot\frac{\mathrm{d}\dot{\boldsymbol{\rho}}}{\mathrm{d}t}\,\mathrm{d}t - \int_0^T M(\boldsymbol{\rho},\boldsymbol{\gamma},\mathbf{u})\dot{\boldsymbol{\rho}}\cdot\mathbf{p}\,\mathrm{d}t = \int_0^T N(\boldsymbol{\rho},\boldsymbol{\gamma},\mathbf{u})\mathbf{h}\cdot\mathbf{p}\,\mathrm{d}t.$$

Summing the two last equalities above yields

$$\dot{\boldsymbol{\rho}}(T)\cdot\mathbf{p}(T) - \dot{\boldsymbol{\rho}}(0)\cdot\mathbf{p}(0) = \int_0^T \mathbf{h}\cdot(N(\boldsymbol{\rho},\boldsymbol{\gamma},\mathbf{u})^\top\mathbf{p})\,\mathrm{d}t.$$

Using this identity with $\mathbf{p}(T)=c$ and $\dot{\boldsymbol{\rho}}(0)=0$ results in Expression (2.11). ■

From this result, we will now state the optimality conditions for Problem ($\mathcal{P}_\theta$). Let us first recall that, according to [36, Proposition I.5.1], the subdifferential of the total variation is well-known, given by

$$\partial\,\mathrm{TV}(\mathbf{u}^*) = \{\boldsymbol{\eta}\in C^0([0,T];\mathbb{R}^{N_r}) \mid \|\boldsymbol{\eta}\|_\infty \leq 1 \text{ and } \int \boldsymbol{\eta}\mathrm{d}\mathbf{u}^* = \mathrm{TV}(\mathbf{u}^*)\}.$$

Let us denote by e_i the i-th vector of the canonical basis of $\mathbb{R}^{N_r}$

Theorem 7. *Let $\mathbf{u}^* = (u_i^*)_{1\leq i\leq N_r}$, denote a solution to Problem ($\mathcal{P}_\theta$), $(\boldsymbol{\rho},\boldsymbol{\gamma})$ the associated solution to (LWR-sd). and let $i\in[\![1,N_r]\!]$. There exists $\boldsymbol{\eta}^*\in\partial\,\mathrm{TV}(\mathbf{u}^*)$ such that*

- *on $\{u_i^*=0\}$, one has $\Psi\cdot e_i\geq 0$,*
- *on $\{u_i^*=1\}$, one has $\Psi\cdot e_i\leq 0$,*
- *on $\{0<u_i^*<1\}$, one has $\Psi\cdot e_i = 0$,*

where the function $\Psi:[0,T]\to\mathbb{R}^{N_r}$ is given by

$$\Psi(t) = N(\boldsymbol{\rho}(t),\boldsymbol{\gamma}(t),\mathbf{u}^*(t))^\top\mathbf{p}^*(t) + \theta_S\left(\sum_{i=1}^{N_r}u_i^*(t) - N_{\max}\right)_+ + \theta_B\boldsymbol{\eta}^*(t)$$

and where $\mathbf{p}^*$ denotes the adjoint state introduced in Proposition 6, associated to the control choice $\mathbf{u}^*$.

Remark 3. *Written in this way, the first order optimality conditions are difficult to use. In the next section, we will introduce an approximation of the total variation of $\mathbf{u}^*$ leading to optimality conditions more easily usable within an algorithm.*

Proof of Theorem 7. To derive the first order optimality conditions for this problem, it is convenient to introduce the so-called indicator function $\iota_{\mathcal{U}_{ad}}$ given by

$$\iota_{\mathcal{U}_{ad}}(\mathbf{u}) = \begin{cases} 0 & \text{if } \mathbf{u}\in\mathcal{U}_{ad} \\ +\infty & \text{else.}\end{cases}$$

Observing that the optimization problem we want to deal with can be recast as

$$\min_{\mathbf{u}\in L^\infty((0,T);\mathbb{R}^{N_r})} \mathcal{J}_\theta(\mathbf{u}) + \iota_{\mathcal{U}_{ad}}(\mathbf{u}),$$

it is standard in nonsmooth analysis to write the first order optimality conditions as:

$$0 \in \partial \left(\mathcal{J}_\theta(\mathbf{u}^*) + \iota_{\mathcal{U}_{ad}}(\mathbf{u}^*) \right),$$

or similarly, by using standard computational rules [36],

$$-\partial C_T(\mathbf{u}^*) - \theta_S \partial S(\mathbf{u}^*) \in \theta_B \partial B(\mathbf{u}^*) + \partial \iota_{\mathcal{U}_{ad}}(\mathbf{u}^*),$$

Let $\mathbf{u} \in \mathcal{U}_{ad}$. The condition above yields the existence of $\boldsymbol{\eta}^* = (\eta_i^*)_{1 \le i \le N_r} \in \partial \, \mathrm{TV}(\mathbf{u}^*)$ such that

$$\mathrm{d}C_T(\mathbf{u}^*)(\mathbf{u} - \mathbf{u}^*) + \theta_S \mathrm{d}S(\mathbf{u}^*)(\mathbf{u} - \mathbf{u}^*) + \theta_B \sum_{i=1}^{N_r} \langle \eta_i^*, u_i - u_i^* \rangle_{L^2(0,T)} \ge 0.$$

Since $\mathbf{u}$ is arbitrary, we infer that for any admissible perturbation $\mathbf{h}$ of $\mathbf{u}^*$ (see Definition 4), one has

$$\mathrm{d}C_T(\mathbf{u}^*)\mathbf{h} + \theta_S \mathrm{d}S(\mathbf{u}^*)\mathbf{h} + \theta_B \sum_{i=1}^{N_r} \langle \eta_i^*, h_i \rangle_{L^2(0,T)} \ge 0,$$

or similarly

$$\int_0^T \mathbf{h} \cdot \left(N(\boldsymbol{\rho}, \boldsymbol{\gamma}, \mathbf{u})^\top \mathbf{p}^* + \theta_S \left(\sum_{i=1}^{N_r} u_i^* - N_{\max} \right)_+ + \theta_B \boldsymbol{\eta}^* \right) \mathrm{d}t \ge 0. \tag{2.15}$$

To analyze this optimality condition, let us introduce the function $\Psi : [0, T] \to \mathbb{R}^{N_r}$ defined by

$$\Psi(t) = N(\boldsymbol{\rho}, \boldsymbol{\gamma}, \mathbf{u})^\top \mathbf{p}^*(t) + \theta_S \left(\sum_{i=1}^{N_r} u_i^* - N_{\max} \right)_+ + \theta_B \boldsymbol{\eta}^*(t). \tag{2.16}$$

In what follows, we will write the optimality conditions holding for the i-th component of $\mathbf{u}^*$, where $i \in [\![1, N_r]\!]$ is given.

Let us assume that the set $\mathcal{I} = \{0 < u_i^* < 1\}$ is of positive Lebesgue measure. Let x_0 be a Lebesgue point of u_i^* in $\mathcal{I}$ and let $(G_n)_{n \in \mathbb{N}}$ be a sequence of measurable subsets with G_n included in $\mathcal{I}$ and containing x_0. Let us consider $\mathbf{h} = (h_j)_{1 \le j \le N_r}$ such that $h_j = 0$ for all $j \in [\![1, N_r]\!] \backslash \{i\}$ and $h_i = \mathbb{1}_{G_n}$. Notice that $\mathbf{u}^* \pm \eta \mathbf{h}$ belongs to $\mathcal{U}_{ad}$ whenever η is small enough. According to (2.15), one has

$$\pm \int_{G_n} \Psi(t) \cdot e_i \, \mathrm{d}t \ge 0.$$

Dividing this inequality by $|G_n|$ and letting G_n shrink to $\{x_0\}$ as $n \to +\infty$ shows that one has

$$\Psi(t) \cdot e_i = 0, \quad \text{a.e. in } \mathcal{I}.$$

Let us now assume that the set $\mathcal{I}_1 = \{u_i^* = 1\}$ is of positive Lebesgue measure. Then, by mimicking the reasoning above, we consider x_1, a Lebesgue point of u_i^* in $\mathcal{I}_1$, and $\mathbf{h} = (h_j)_{1 \le j \le N_r}$ such that $h_j = 0$ for all $j \in [\![1, N_r]\!] \backslash \{i\}$ and $h_i = -\mathbb{1}_{G_n}$, where $(G_n)_{n \in \mathbb{N}}$ is a sequence of measurable subsets with G_n included in $\mathcal{I}_1$

and containing x_1. According to (2.15), one has

$$-\int_{G_n} \Psi(t) \cdot e_i \, \mathrm{d}t \geq 0.$$

As above, we divide this inequality by $|G_n|$ and let G_n shrink to $\{x_1\}$ as $n \to +\infty$. We recover that $\Psi(t) \cdot e_i \leq 0$.

Regarding now the set $\mathcal{I}_0 = \{u_i^* = 0\}$, the reasoning is a direct adaptation of the case above, which concludes the proof. ■

2.4 Towards a numerical algorithm

In this section, we introduce an exploitable approximation of the problem we are dealing with and describe the algorithm implemented in the numerical part.

2.4.1 An approximate version of Problem ($\mathcal{P}_\theta$)

The fact that Problem ($\mathcal{P}_\theta$) involves the total variation of control makes the problem non-differentiable. Of course, dedicated algorithms exist to take into account such a term in the solution, for instance Chambolle's projection algorithm [16]. Nevertheless, in order to avoid too costly numerical approaches, we have chosen to consider a simple differentiable approximation of the term $B(\mathbf{u})$, namely

$$B_\nu(\mathbf{u}) = \sum_{i=1}^{N_r} \mathrm{TV}_\nu(u_i), \tag{2.17}$$

where $\nu > 0$ is a small parameter and the total variation $\mathrm{TV}(u_i)$ is approximated by a differentiable functional in $L^2(0,T)$, denoted $\mathrm{TV}_\nu(u_i)$, where $\nu > 0$ stands for a regularization parameter. The concrete choice of the differentiable approximation of the TV standard will be discussed in the rest of this section. We will also give some elements on its practical implementation.

We are thus led to consider the following approximate version of Problem ($\mathcal{P}_\theta$):

$$\boxed{\inf_{\mathbf{u} \in \mathcal{U}_{ad}} \mathcal{J}_{\theta,\nu}(\mathbf{u})}. \tag{$\mathcal{P}_{\theta,\nu}$}$$

where $\mathcal{U}_{ad}$ is given by (2.5), and $\mathcal{J}_{\theta,\nu}$ is given by

$$\mathcal{J}_{\theta,\nu}(\mathbf{u}) = C_T(\mathbf{u}) + \theta_S S(\mathbf{u}) + \theta_B B_\nu(\mathbf{u}), \tag{2.18}$$

We will see that this approximation is in fact well adapted to a practical use. Indeed, the first order optimality conditions for this approximated problem can be rewritten in a very concise and workable way, unlike the result stated in Theorem 7. This is the purpose of the following result.

Theorem 8. *Let $\mathbf{u}^* \in \mathcal{U}_{ad}$ denote a local minimizer for Problem ($\mathcal{P}_{\theta,\nu}$) and $(\boldsymbol{\rho}, \boldsymbol{\gamma})$ the associated solution to* (LWR-sd)*. Then, $\mathbf{u}^*$ satisfies the first order necessary condition*

$$\Lambda(\cdot) = 0 \quad \textit{a.e. on } [0,T],$$

where $\Lambda : [0,T] \to \mathbb{R}^{N_r}$ *is defined by*

$$\Lambda(t) = \min\left\{\mathbf{u}^*(t), \max\left\{\mathbf{u}^*(t) - 1, \nabla_{\mathbf{u}}\mathcal{J}_{\theta,\nu}(\mathbf{u}^*)(t)\right\}\right\},$$

and

$$\nabla_{\mathbf{u}}\mathcal{J}_{\theta,\nu}(\mathbf{u}^*) : [0,T] \ni t \mapsto N(\boldsymbol{\rho}(t), \boldsymbol{\gamma}(t), \mathbf{u}(t))^\top \mathbf{p}^*(t) + \theta_S \left(\sum_{i=1}^{N_r} u_i^*(t) - N_{\max}\right)_+ + \theta_B \sum_{i=1}^{N_r} \nabla_{\mathbf{u}} \mathrm{TV}_\nu(u_i)(t),$$

where $\mathbf{p}^*$ *has been introduced in Theorem 7, the min, max operations being understood componentwise, and the term* $\nabla_{\mathbf{u}}$ *denoting the gradient with respect to* $\mathbf{u}$ *in* $L^2(0,T)$.

Proof. The proof is similar to the proof of Theorem 7. Indeed, let $\mathbf{u}^*$ be a local minimizer for Problem ($\mathcal{P}_{\theta,\nu}$). The first order optimality conditions are given by the so-called Euler inequation and read $\mathrm{d}\mathcal{J}_{\theta,\nu}(\mathbf{u}^*)\mathbf{h} \geq 0$, or similarly

$$\int_0^T \nabla_{\mathbf{u}}\mathcal{J}_{\theta,\nu}(\mathbf{u}^*) \cdot \mathbf{h}\, \mathrm{d}t \geq 0.$$

for every admissible perturbation $\mathbf{h}$, as defined in Definition 4. Let us fix $i \in [\![1, N_r]\!]$. By mimicking the reasoning involving Lebesgue points in the proof of Theorem 7, we get

- on $\{u_i^* = 0\}$, one has $\nabla_{\mathbf{u}}\mathcal{J}_{\theta,\nu}(\mathbf{u}^*) \cdot e_i \geq 0$,
- on $\{u_i^* = 1\}$, one has $\nabla_{\mathbf{u}}\mathcal{J}_{\theta,\nu}(\mathbf{u}^*) \cdot e_i \leq 0$,
- on $\{0 < u_i^* < 1\}$, one has $\nabla_{\mathbf{u}}\mathcal{J}_{\theta,\nu}(\mathbf{u}^*) \cdot e_i = 0$.

Note that, on $\{u_i^* = 0\}$, one has $\nabla_{\mathbf{u}}\mathcal{J}_{\theta,\nu}(\mathbf{u}^*) \cdot e_i \geq 0$ and then $\Lambda(t) \cdot e_i = \min\{0, \max\{-1, \nabla_{\mathbf{u}}\mathcal{J}_{\theta,\nu}(\mathbf{u}^*)(t) \cdot e_i\}\} = 0$. On $\{u_i^* = 1\}$, one has $\nabla_{\mathbf{u}}\mathcal{J}_{\theta,\nu}(\mathbf{u}^*) \cdot e_i \leq 0$ and therefore $\Lambda(t) \cdot e_i = \min\{1, \max\{0, \nabla_{\mathbf{u}}\mathcal{J}_{\theta,\nu}(\mathbf{u}^*)(t) \cdot e_i\}\} = 0$. On $\{0 < u_i^* < 1\}$, one has $\nabla_{\mathbf{u}}\mathcal{J}_{\theta,\nu}(\mathbf{u}^*) \cdot e_i = 0$ and therefore $\Lambda(t) \cdot e_i = \min\{u_i^*, \max\{u_i^*(t) - 1, 0\}\} = 0$.

Conversely, let us assume that $\Lambda(\cdot) = 0$. On $\{u_i^* = 0\}$, one has

$$0 = \Lambda(t) \cdot e_i = \min\{0, \max\{-1, \nabla_{\mathbf{u}}\mathcal{J}_{\theta,\nu}(\mathbf{u}^*)(t) \cdot e_i\}\},$$

so that $\max\{-1, \nabla_{\mathbf{u}}\mathcal{J}_{\theta,\nu}(\mathbf{u}^*)(t) \cdot e_i\} \geq 0$ and finally, $\nabla_{\mathbf{u}}\mathcal{J}_{\theta,\nu}(\mathbf{u}^*)(t) \cdot e_i \geq 0$. A similar reasoning yields the optimality conditions on $\{u_i^* = 1\}$ and $\{0 < u_i^* < 1\}$. The conclusion follows. ■

Practical computation of the TV operator gradient in a discrete framework. From a practical point of view, we discretize the space of controls, which leads us to consider a time discretization denoted $(t_n)_{1 \leq n \leq N_T}$ with $N_T \in \mathbb{N}^*$ fixed, as well as piecewise constant controls in time, denoted $(u_i^n)_{\substack{1 \leq i \leq N_r \\ 1 \leq n \leq N_T}}$.

Hence, the term u_i^n corresponds to the control at road i and time n. The simplest discrete version of the TV semi-norm is given by

$$\mathrm{TV}(\mathbf{u}) \simeq \sum_{i=1}^{N_r} \sum_{n=2}^{N_T} |u_i^n - u_i^{n-1}|.$$

In order to manipulate differentiable expressions, we introduce for some $\nu > 0$, the smoothed discrete TV operator is given by

$$\mathrm{TV}_\nu : \mathbb{R}^{N_r \times N_T} \ni (u_i^n)_{\substack{1\le i\le N_r \\ 1\le n\le N_T}} \mapsto \sum_{i=1}^{N_r}\sum_{n=2}^{N_T} a_\nu(u_i^n - u_i^{n-1}), \tag{2.19}$$

where $a_\nu : \mathbb{R} \ni x \mapsto \sqrt{x^2+\nu^2}$.

Let $(i_0, n_0) \in [\![1, N_r]\!] \times [\![1, N_T]\!]$. In what follows, we will use a discrete equivalent of Theorem 8, involving the gradient of TV_ν, obtained from the expression

$$\partial_{u_{i_0}^{n_0}} \mathrm{TV}_\nu(\mathbf{u}) = \begin{cases} a_\nu'(u_{i_0}^{n_0} - u_{i_0}^{n_0-1}) - a_\nu'(u_{i_0}^{n_0+1} - u_{i_0}^{n_0}) & \text{if } 2 \le n_0 \le N_T - 1 \\ a_\nu'(u_{i_0}^{N_T} - u_{i_0}^{N_T-1}) & \text{if } n_0 = N_T \\ -a_\nu'(u_{i_0}^{2} - u_{i_0}^{1}) & \text{if } n_0 = 1, \end{cases}$$

where $\mathbf{u} = (u_i^n)_{\substack{1\le i\le N_r \\ 1\le n\le N_T}}$ is given

2.4.2 Numerical solving of the primal and dual problems

The models we use are already discretized in space. We now explain how we discretize them in time. We recall that, at the end of section 2.4.1, we have already considered that the controls are assimilated to piecewise constant functions on each cell of the considered mesh and on each time step.

Finite volumes scheme for the primal problem. The main transport equation on network is solved by integrating for each road the finite volume flow given by (LWR-sd) with an explicit Euler scheme:

$$\rho_{i,j}^{n+1} = \rho_{i,j}^n - \frac{\triangle t_n}{\triangle x_{i,j}}\left(\mathcal{F}_{i,j+\frac{1}{2}}^n - \mathcal{F}_{i,j-\frac{1}{2}}^n\right), \quad 1 \le i \le N_r, \quad 1 \le j \le N_c,$$

using the local-Lax numerical flux

$$\mathcal{F}_{i,j+\frac{1}{2}}^n = \mathcal{F}(\rho_{i,j}^n, \rho_{i,j+1}^n), \quad 2 \le j \le N_c - 1,$$

and the Neumann boundary conditions

$$\mathcal{F}_{i,\frac{1}{2}}^n := \gamma_{i,L}^n, \quad \mathcal{F}_{i,N_c+\frac{1}{2}}^n := \gamma_{i,R}^n, \quad 1 \le i \le N_r,$$

where $\gamma_{i,L}^n$ and $\gamma_{i,R}^n$ are obtained as the solution to the linear programming system

$$\boldsymbol{\gamma}^n = \phi^{\mathrm{LP}}(\boldsymbol{\rho}^n, \mathbf{u}^n).$$

Euler scheme for adjoint problem. To solve the backward ODE (2.12), it is convenient to introduce $Z(t) := (M(T-t))^T$, and $\mathbf{q}(t) := \mathbf{p}(T-t)$ such that we are now dealing with the Cauchy system:

$$\begin{cases} \mathbf{q}'(t) = Z(t)\,\mathbf{q}(t), \quad t \in (0, T), \\ \mathbf{q}(0) = \mathbf{c}, \end{cases}$$

that we integrate with a classical explicit Euler scheme:

$$\mathbf{q}^1 = \mathbf{c}, \quad \mathbf{q}^{n+1} = (I + \triangle t_n \, Z^n)\, \mathbf{q}^n, \quad n = 1, \cdots, N_T - 1.$$

The solution is finally recovered using that $\mathbf{p}^n = \mathbf{q}^{N_T - n + 1}$.

2.4.3 Optimization algorithms

The starting point of the algorithm we implement is based on a standard primal-dual approach, in which the state and the adjoint are computed in order to deduce the gradient of the considered functional. We combine it with a projection method in order to guarantee the respect of the L^∞ constraints on the control. This method has the advantage of being robust, as it generally allows a significant decrease of the cost functional. On the other hand, it often has the disadvantage of being very local, which results in an important dependence on the initialization. Moreover, one can expect that there are many local minima, since the targeted problem is intrinsically of infinite dimension, which makes the search difficult.

We will propose a modification of this well-known method, using a fixed point method inspired by the optimality condition stated in Theorem 8.

Projected gradient descent

A direct approach is to consider the gradient algorithm, in which we deal with the condition $0 \leq u_i^k \leq 1$ by projection, according to Algorithm 1. The specific difficulty of this approach is to find a suitable descent-step δ_k, which must be small enough to ensure descent but large enough for the algorithm to converge in a reasonable number of iterations. We hereby combine this algorithm with a scheduler Equation 2.20 inspired from classical learning rate scheduler in deep-learning [97] to select an acceptable descent step, where δ_0 and decay are given real numbers.

$$\delta_k := \frac{\delta_0}{1 + \mathrm{decay} \times k}. \tag{2.20}$$

Algorithm 1 Optimal control by projected gradient descent method (GD)

Require: $\boldsymbol{\rho}^0, \mathbf{u}^0, \mathrm{tol} > 0, \mathrm{itermax} > 0$
  Initialization: $k = 0$
  while $||\mathbf{u}^{k+1} - \mathbf{u}^k|| > \mathrm{tol}$ **and** $k \leq \mathrm{itermax}$ **do**
    $(\boldsymbol{\rho}, \boldsymbol{\gamma}) \leftarrow$ solution to $\begin{cases} \boldsymbol{\rho}' = f^{\mathrm{FV}}(\boldsymbol{\rho}, \boldsymbol{\gamma}) \\ \boldsymbol{\gamma} = \phi^{\mathrm{LP}}(\boldsymbol{\rho}, \mathbf{u}^k) \\ \boldsymbol{\rho}(0) = \boldsymbol{\rho}^0 \end{cases}$
    $\mathbf{p} \leftarrow$ solution to $\begin{cases} \mathbf{p}' + M(\boldsymbol{\rho}, \boldsymbol{\gamma}, \mathbf{u}^k)^T \mathbf{p} = \mathbf{0} \\ \mathbf{p}(T) = \mathbf{c} \end{cases}$
    $\mathbf{u}^{k+1} \leftarrow \mathrm{proj}_{[0,1]}(\mathbf{u}^k - \delta_k \nabla \mathcal{J}_{\theta,\nu}(\mathbf{u}^k))$, with $\delta_k > 0$ such that $\mathcal{J}_{\theta,\nu}(\mathbf{u}^{k+1}) \leq \mathcal{J}_{\theta,\nu}(\mathbf{u}^k)$
    $k \leftarrow k + 1$
  end while

Fixed point method A fixed-point (FP) algorithm is derived using the first-order optimality conditions stated in Theorems 7 and 8, by rewriting them in a fixed-point formulation. We have seen that they write under the form

$$u_i \in I_\mu \Rightarrow \partial_{u_i} \mathcal{J}_{\theta,\nu}(\mathbf{u}) \in E_\mu,$$

where $\mu \in \{0, *, 1\}$, $I_0 = \{0\}$, $I_* = (0,1)$, $I_1 = \{1\}$, $E_0 = \mathbb{R}_+$, $E_* = \{0\}$, $E_1 = \mathbb{R}_-$. This rewrites as

$$u_i \in I_\mu \Rightarrow u_i \in F_\mu := \{u_i | \partial_{u_i} \mathcal{J}_{\theta,\nu}(\mathbf{u}) \in E_\mu\}.$$

This leads us to compute u_i by using the following fixed-point relationship

$$u_i^{k+1} = 1 \times \mathbb{1}_{\{\partial_{u_i} \mathcal{J}_{\theta,\nu}(\mathbf{u}^k) < -\kappa\}} + u_i^k \, \mathbb{1}_{\{-\kappa \leq \partial_{u_i} \mathcal{J}_{\theta,\nu}(\mathbf{u}^k) \leq \kappa\}} + 0 \times \mathbb{1}_{\{\partial_{u_i} \mathcal{J}_{\theta,\nu}(\mathbf{u}^k) > \kappa\}}, \tag{2.21}$$

where $\kappa > 0$ is a small threshold preventing too small gradients from modifying the control. This finally leads us to Algorithm 2.

Remark 4. *The advantage of the fixed-point formulation can be illustrated by the fact that some common control configurations can lead to a flat gradient while optimisation is still possible, i.e. gradient descent is too local to detect certain appropriate (and existing) descent direction. Let's construct such a case where the fixed-point algorithm has a major advantage over gradient descent. We recall the constraints of the linear programming for the* 2×2 *case:*

$$\begin{aligned}
0 &\leq \gamma_1^R \leq \gamma_1^{R,\max}, && (2.22)\\
0 &\leq \gamma_2^R \leq \gamma_2^{R,\max}, && (2.23)\\
0 &\leq \alpha_{\boldsymbol{u}} \gamma_1^R + \beta_{\boldsymbol{u}} \gamma_2^R \leq (1-u_3)\gamma_3^{L,\max}, && (2.24)\\
0 &\leq (1-\alpha_{\boldsymbol{u}})\gamma_1^R + (1-\beta_{\boldsymbol{u}})\gamma_2^R \leq (1-u_4)\gamma_4^{L,\max}. && (2.25)
\end{aligned}$$

Considering for instance a case where $u_3 = 1$*, then Equation 2.24 yields* $\alpha_{\boldsymbol{u}} \gamma_1^R + \beta_{\boldsymbol{u}} \gamma_2^R = 0$*. By positivity, this implies that* $\alpha_{\boldsymbol{u}} \gamma_1^R = \beta_{\boldsymbol{u}} \gamma_2^R = 0$*. If furthermore* $u_4 > 0$*, then it follows that* $\alpha_{\boldsymbol{u}} = P_\varepsilon^{\bar{\alpha}}(1-u_4) \neq 0$ *and* $\beta_{\boldsymbol{u}} = P_\varepsilon^{\bar{\beta}}(1-u_4) \neq 0$ *(see (2.28) in Appendix for the definition of these functions). It remains that necessarily,* $\gamma_1^R = \gamma_2^R = 0$ *and the junction is blocked. The point here is that the only way to influence the network – therefore acting on the value of* $\mathcal{J}_{\theta,\nu}$ *– is by having a control such that* γ_1^R *or* γ_2^R *becomes positive, and this can be obtained only when* u_4 *is set to 0. Indeed, we would now have* $\alpha_{\boldsymbol{u}} = P_\varepsilon^{\bar{\alpha}}(1) = P_\varepsilon^{\bar{\beta}}(1) = \beta_{\boldsymbol{u}} = 0$*, releasing the constraint given by Equation 2.24 on the values of* γ_1^R, γ_2^R*.*

However, if the algorithm finds a perturbation $\delta u_4 > 0$ *small enough such that* $u_4 - \delta u_4 \neq 0$*, it would yields* $\mathcal{J}_{\theta,\nu}(\mathbf{u}) = \mathcal{J}_{\theta,\nu}(\mathbf{u} - \delta u_4 e_4)$*, thus* $\partial_{u_4} \mathcal{J}_{\theta,\nu}(\mathbf{u}) = 0$*. As a result, the gradient descent step is stationnary for* u_4 *even though the perturbation* δu_4 *was in the good direction in order to decrease* $\mathcal{J}_{\theta,\nu}$ *after several more iterations.*

With the fixed-point algorithm instead, we find the suitable control u_4 *in one iteration from the previous configuration thanks to the indicator functions, since the sign of the gradient contains all the information that we needed. This allows us to circumvent the threshold phenomenon of the gradient described above.*

Algorithm 2 Optimal control with Fixed Point method (FP)

Require: $\boldsymbol{\rho}^0, \mathbf{u}^0, \text{tol} > 0, \text{itermax} > 0$
  Initialization: $k = 0$
  while $||\mathbf{u}^{k+1} - \mathbf{u}^k|| > \text{tol}$ **and** $k \leq \text{itermax}$ **do**

$$(\boldsymbol{\rho}, \boldsymbol{\gamma}) \leftarrow \text{solution to } \begin{cases} \boldsymbol{\rho}' = f^{\text{FV}}(\boldsymbol{\rho}, \boldsymbol{\gamma}) \\ \boldsymbol{\gamma} = \phi^{\text{LP}}(\boldsymbol{\rho}, \mathbf{u}^k) \\ \boldsymbol{\rho}(0) = \boldsymbol{\rho}^0 \end{cases}$$

$$\mathbf{p} \leftarrow \text{solution to } \begin{cases} \mathbf{p}' + M(\boldsymbol{\rho}, \boldsymbol{\gamma}, \mathbf{u}^k)^T \mathbf{p} = \mathbf{0} \\ \mathbf{p}(T) = \mathbf{c} \end{cases}$$

$$\mathbf{u}^{k+1} \leftarrow \mathbb{1}_{\{\nabla \mathcal{J}_{\theta,\nu}(\mathbf{u}^k) < -\kappa\}} + \mathbf{u}^k \mathbb{1}_{\{-\kappa \leq \nabla \mathcal{J}_{\theta,\nu}(\mathbf{u}^k) \leq \kappa\}} + 0 \times \mathbb{1}_{\{\partial_{u_i} \mathcal{J}_{\theta,\nu}(\mathbf{u}^k) > \kappa\}}$$

  $k \leftarrow k + 1$
  end while

Hybrid GDFP methods

Since gradient descent theoretically always guarantees a direction of descent (provided we choose a small enough step) but can be slow and/or remain trapped in a local extremum (see Remark 4), and since the fixed point appears more exploratory but does not guarantee descent, it seems worthwhile to investigate the hybridization of both methods.

The chosen algorithm is implemented by computing most of the iterations by gradient descent and using the fixed point method every $K \in \mathbb{N}$ iterations in the expectation of escaping from possible basins of attraction of local minimizers. Note that K is a hyper-parameter of the method. This is summarized in Algorithm 3.

Instead of performing FP steps regularly, we can choose to space them in order to have more more iterations for the gradient descent to converge. This second version is given in Algorithm 4 where the time between two FP steps increases by a factor of τ.

Algorithm 3 Optimal control with hybrid GD & FP algorithm (GDFP)

Require: $\boldsymbol{\rho}^0, \mathbf{u}^0, \text{tol} > 0, \text{itermax} > 0$
  Initialization: $k = 0$
  while $||\mathbf{u}^{k+1} - \mathbf{u}^k|| > \text{tol}$ **and** $k \leq \text{itermax}$ **do**

$$(\boldsymbol{\rho}, \boldsymbol{\gamma}) \leftarrow \text{solution to } \begin{cases} \boldsymbol{\rho}' = f^{\text{FV}}(\boldsymbol{\rho}, \boldsymbol{\gamma}) \\ \boldsymbol{\gamma} = \phi^{\text{LP}}(\boldsymbol{\rho}, \mathbf{u}^k) \\ \boldsymbol{\rho}(0) = \boldsymbol{\rho}^0 \end{cases}$$

$$\mathbf{p} \leftarrow \text{solution to } \begin{cases} \mathbf{p}' + M(\boldsymbol{\rho}, \boldsymbol{\gamma}, \mathbf{u}^k)^T \mathbf{p} = \mathbf{0} \\ \mathbf{p}(T) = \mathbf{c} \end{cases}$$

    if $k\%K == 0$ **then**

$$\mathbf{u}^{k+1} \leftarrow \mathbb{1}_{\{\nabla \mathcal{J}_{\theta,\nu}(\mathbf{u}^k) < -\kappa\}} + \mathbf{u}^k \mathbb{1}_{\{-\kappa \leq \nabla \mathcal{J}_{\theta,\nu}(\mathbf{u}^k) \leq \kappa\}} + 0 \times \mathbb{1}_{\{\partial_{u_i} \mathcal{J}_{\theta,\nu}(\mathbf{u}^k) > \kappa\}}$$

    else

$$\mathbf{u}^{k+1} \leftarrow \text{proj}_{[0,1]}(\mathbf{u}^k - \delta_k \nabla \mathcal{J}_{\theta,\nu}(\mathbf{u}^k)), \text{ with } \delta_k > 0 \text{ such that } \mathcal{J}_{\theta,\nu}(\mathbf{u}^{k+1}) \leq \mathcal{J}_{\theta,\nu}(\mathbf{u}^k)$$

    end if
    $k \leftarrow k + 1$
  end while

Algorithm 4 Optimal control with GDFP with spaced FP steps

Require: $\boldsymbol{\rho}^0, \mathbf{u}^0, tol > 0, \text{itermax} > 0, K_0, \tau$
  Initialization: $k = 0, K = K_0$
  while $||\mathbf{u}^{k+1} - \mathbf{u}^k|| > tol$ **and** $k \leq \text{itermax}$ **do**
    $(\boldsymbol{\rho}, \boldsymbol{\gamma}) \leftarrow$ solution of $\begin{cases} \boldsymbol{\rho}' = f^{\text{FV}}(\boldsymbol{\rho}, \boldsymbol{\gamma}) \\ \boldsymbol{\gamma} = \phi^{\text{LP}}(\boldsymbol{\rho}, \mathbf{u}^k) \\ \boldsymbol{\rho}(0) = \boldsymbol{\rho}^0 \end{cases}$
    $\mathbf{p} \leftarrow$ solution of $\begin{cases} \mathbf{p}'(t) + M^T \mathbf{p} = \mathbf{0} \\ \mathbf{p}(T) = \mathbf{c} \end{cases}$
    if $k\%K == 0$ **then**
      $\mathbf{u}^{k+1} \leftarrow \mathbb{1}_{\{\nabla \mathcal{J}(\mathbf{u}^k) < -\kappa\}} + \mathbf{u}^k \mathbb{1}_{\{-\kappa \leq \nabla \mathcal{J}(\mathbf{u}^k) \leq \kappa\}} + 0 \times \mathbb{1}_{\{\partial_{u_i} \mathcal{J}_{\theta,\nu}(\mathbf{u}^k) > \kappa\}}$
      $K \leftarrow \tau K$
    else
      $\mathbf{u}^{k+1} \leftarrow \text{proj}_{[0,1]}(\mathbf{u}^k - \delta_k \nabla \mathcal{J}(\mathbf{u}^k))$, with $\delta_k > 0$ such that $\mathcal{J}(\mathbf{u}^{k+1}) \leq \mathcal{J}(\mathbf{u}^k)$
    end if
    $k \leftarrow k + 1$
  end while

2.5 Numerical Results

This section presents some results obtained in various situations using the methods presented above. In all that follows, we assume that $\rho^{\max} = 1$, $v^{\max} = 1$ and $L = 1$. We will first validate our approach on single junctions before studying more complex road networks. These are namely: a traffic circle and a three lanes network of intermediate size with a configuration unfavorable to our objective.

2.5.1 Single junctions

We start by considering single junctions of type 1×1, 1×2, 2×1 and 2×2 as they will be the building blocks of the more complex networks.

We consider 50 mesh cells per road and the initial density equals 0.66. The route to empty is always composed of an incoming and an outgoing road. The time interval of the simulations is adjusted in order to allow the route to be completely emptied: the final time T thus equals respectively 6, 3.5, 10 and 5 for the four junctions. We start the optimization algorithms with initial controls equal to 0 and set the convergence threshold to having $\|\Lambda\|$ less than 10^{-1}, with a prescribed maximum of 100 iterations. Neither constraints on the number of controls ($\theta_S = 0$) nor BV regularization ($\theta_B = 0$) are considered for these test cases.

Figure 2.1 shows us the comparison between the cost functional history when using the gradient descent (GD), the fixed-point (FP) and the hybrid (GDFP) methods. Specific numerical parameters of the algorithm are given in Table 2.1. We observe fundamental differences in behaviour between the cost functionals obtained by GD and the one obtained by FP. Indeed, GD allows a regular descent whereas FP generates jumps and oscillations, which allows a better exploration of the parameters and avoids certain unsatisfactory local minima. On Figure 2.2, we also plotted the evolution of the optimality conditions Λ for all methods: as expected, we observe that this quantity reaches values close to 0 at the optimal point.

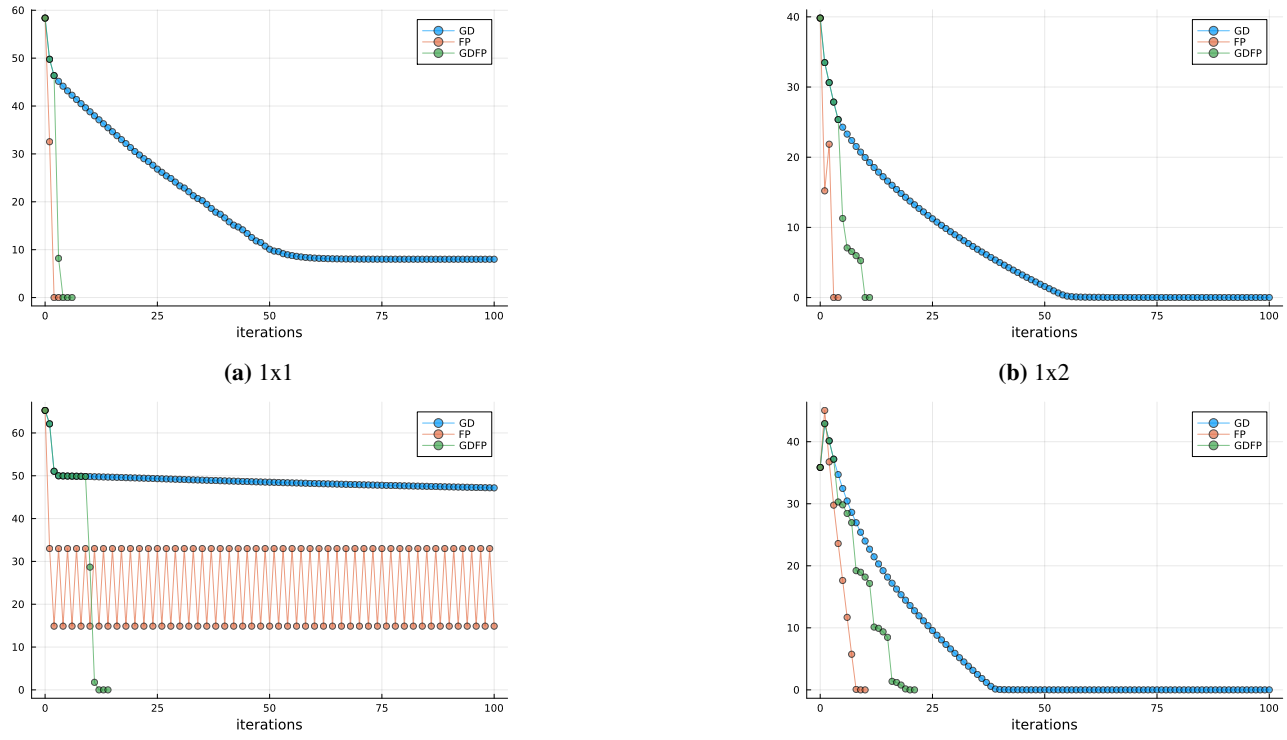
The obtained controls are shown in Figure 2.3 and seem relevant. For instance, in the 2×1 case, the control of the outgoing road 3 is fully activated only after time $t \approx 8$ to enable the emptying of the incoming road 1 first.

We further note that the controls are essentially sparse and non-oscillating. They are almost bang-bang, i.e. take only values 0 and 1.

These results seem to suggest that the hybrid method is the most likely to generalize to larger graphs because of its ability to explore and find critical points while still being able to provide convergence. It is therefore the one we will use in the following.

Symbol	Name	1x1	1x2	2x1	2x2
K	FP trigger in GDFP	3	5	10	2
κ	vanishing gradient threshold in FP	0	10^{-10}	0	10^{-1}
δ_0	initial descent step	1	5×10^{-2}	2×10^{-1}	10^{-1}
decay	decay in scheduler	10^{-2}	10^{-1}	10^{-1}	10^{-1}

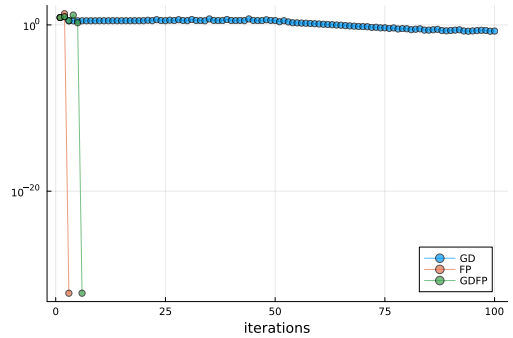
Table 2.1: Numerical parameters for single junctions.



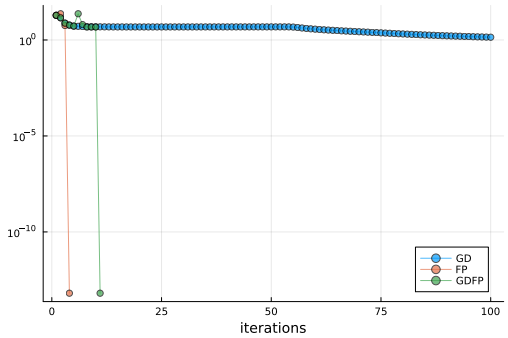
(a) 1x1 **(b)** 1x2

(c) 2x1 **(d)** 2x2

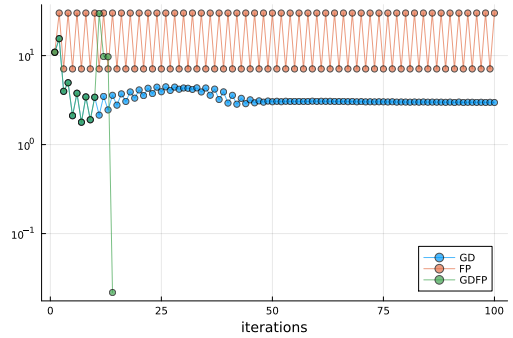
Figure 2.1: (Single junctions) Cost functional as function of iterations for the gradient descent (GD), the fixed point (FP) and the hybrid (GDFP) methods.



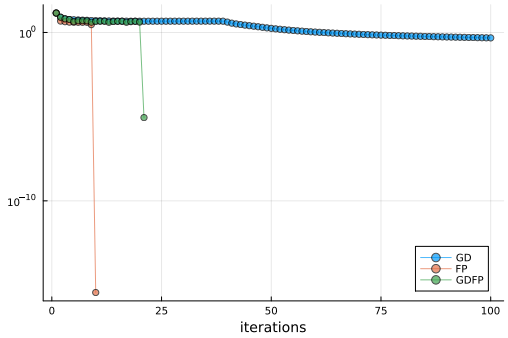
(a) 1x1



(b) 1x2

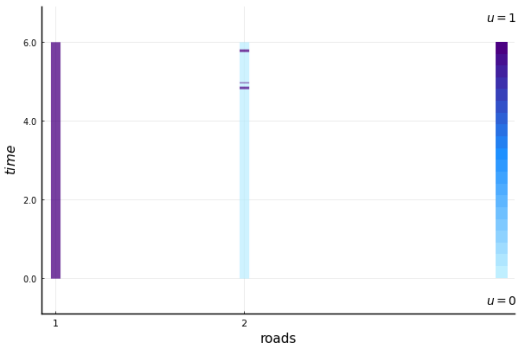


(c) 2x1

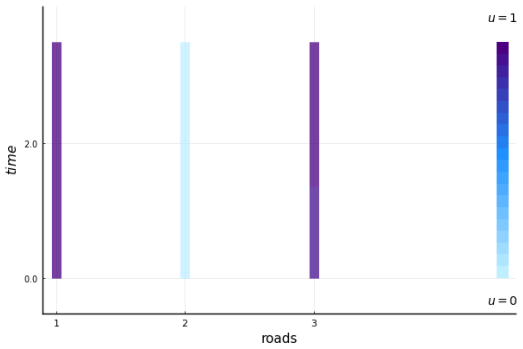


(d) 2x2

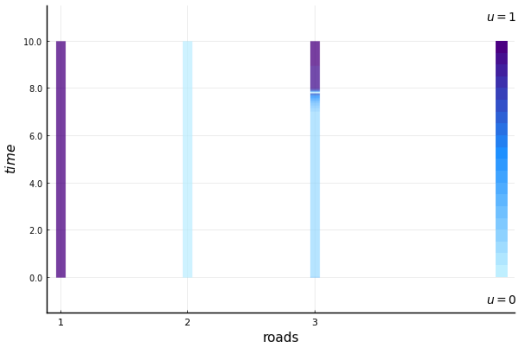
Figure 2.2: (Single junctions) Iterations of the convergence criterion $||\Lambda||$.



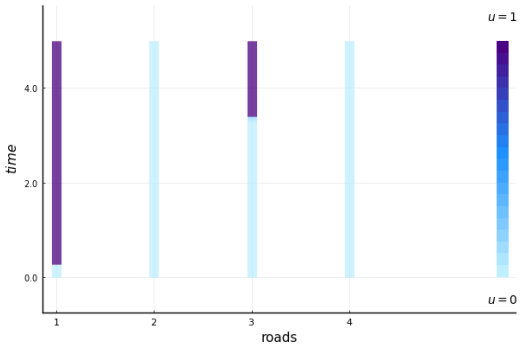
(a) 1x1



(b) 1x2



(c) 2x1



(d) 2x2

Figure 2.3: (Single junctions) Optimal controls obtained with the GDFP method. The routes to empty are respectively $(1, 2)$, $(1, 3)$, $(1, 3)$, $(1, 3)$.

(a) Road network and route to empty in red.

(b) Density at final time without control.

Figure 2.4: (Traffic circle) Left: Traffic circle configuration. The roads $(2, 3, 5, 7)$ in the circle are counter-clockwise, roads $(1, 6)$ are incoming and roads $(4, 8)$ are outgoing. The route to empty $(1, 2, 3, 5, 8)$ is in red. Right: Numerical simulations without control at time $T = 10$.

2.5.2 Traffic circle

We consider the traffic circle test case as proposed in [78]. It is composed of 8 roads as depicted in Fig. **??** including 2 incoming roads, 2 outgoing roads and the 4 circle roads. The roads $(2, 3, 5, 7)$ in the circle are counter-clockwise, roads $(1, 6)$ are incoming and roads $(4, 8)$ are outgoing. Initial density is taken constant equal to 0.66 on each road and, in Fig. 2.4, we observe that congestion appears if no control is applied.

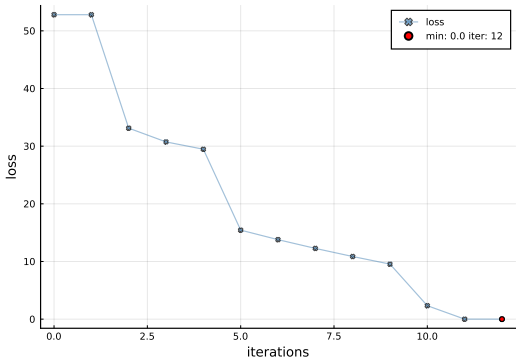
The route to evacuate is chosen to be $(1, 2, 3, 5, 8)$ (see Figure 2.4a) and there is no constraints on the maximal number of controls ($\theta_S = 0$) or BV regularization ($\theta_B = 0$). The parameters are given in Table 2.2.

Results are gathered on Figure 2.5. On the upper left panel, the cost functional history is depicted. The first four iterations of the gradient descent make the cost functional decrease very slowly, except for the second iteration, and then a FP step triggers the jump seen at iteration 5. A few GD steps are then observed and then another FP step results in a second jump at iteration 10. Then the gradient descent is able to reach a satisfactory local minimum by the 12th iteration. We note that the optimality function Λ (Fig. 2.5) follows essentially the same behaviour and reaches a small value at the final iteration.

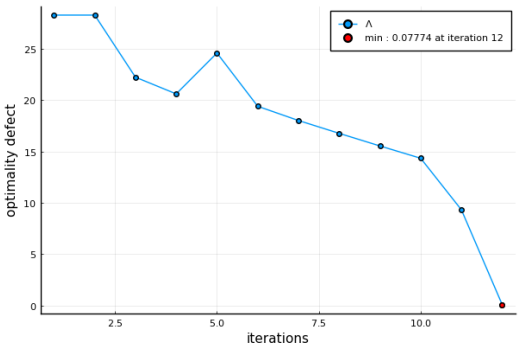
The control obtained by the algorithm is given in Figure 2.5c, bottom left. We observe first that roads 1 and 6 are allways controlled since they are entry roads on the network and would add new vehicles, and then that road 3 entrance is always controlled to drive the flow to the outgoing road 4. Furthermore, while road 5 entrance is mostly controlled after time $t = 8$ to let cars from road 3 leave the route before, control on road 7 entrance is quite the opposite: up to time $t = 8$, it is activated to let the flow circulate as road 8 entrance is open, then from time $t = 8$ to $t = 10$, it is deactivated as outgoing road 8 entrance is now closed. Finally, on Figure 2.5d, we can check that the route is effectively empty at final time.

Parameters	$\boldsymbol{\rho}_0$	$\mathbf{u}_0$	N_c	κ	tol	K_0	τ	T	δ_0	decay
Traffic circle	0.66	1	20	10^{-6}	10^{-2}	5	2	10	5×10^{-2}	10^{-1}
Three lanes network	0.66	1	5	10^{-3}	10^{-2}	5	3	30	1	10^{-2}

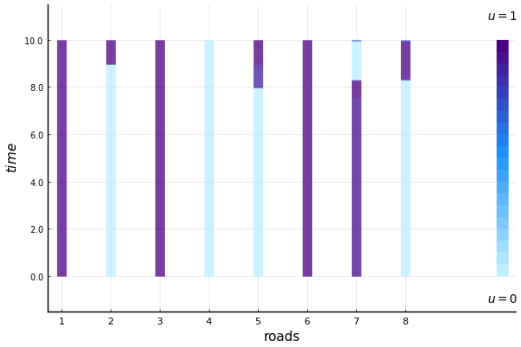
Table 2.2: Numerical parameters for the *traffic circle* and the *three lanes* networks.



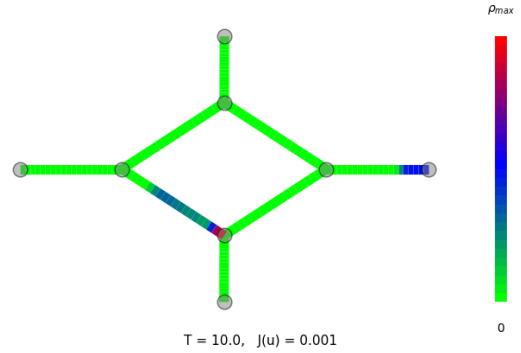
(a) Cost functional history



(b) $||\Lambda||_2$ history



(c) Control



(d) Density at final time

Figure 2.5: (Traffic circle). Results obtained with GDFP algorithm with spaced FP steps.

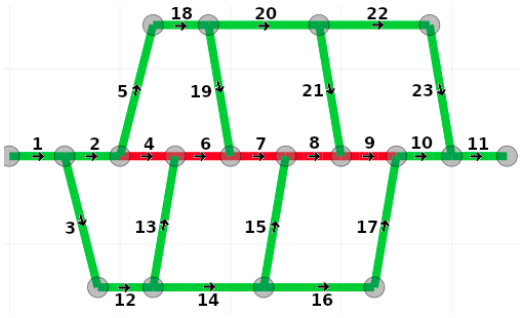
2.5.3 Three lanes network

We consider the three lanes network composed of 23 roads as shown in Fig. 2.6: all the roads are directed from left to right, with road 1 and road 11 being respectively the incoming and outgoig leaves. Initial density is sill constant equal to 0.66 and a corresponding incoming flux is imposed at the entrance of the network on road 1. Neumann boundary conditions are used at the end of the outgoing road 11. The traffic distribution at each junction are uniform: there are no preferred trajectories. Fig. **??** depicts the result of the simulation without control: the density in the central lane tends to be saturated as several roads lead onto it.

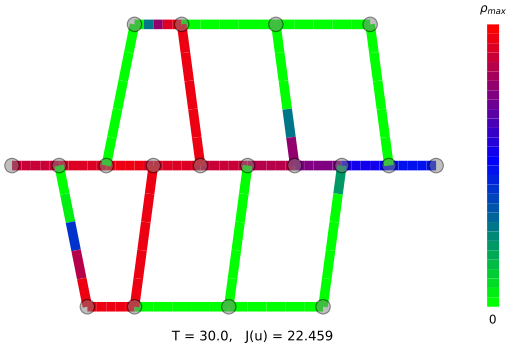
In the sequel, we would like to determine an optimal control in order to empty the central lane, made of roads $(4, 6, 7, 8, 9)$ (see Figure 2.6a), with $N_{max} = 5$. Note that, contrary to the previous test cases, controlling this route may not prevent the flow to circulate from the ingoing to the outgoing road.

First we consider the optimal problem with essentially no limitation on the number of active controls ($\theta_S = 10^{-8}$) nor BV regularization ($\theta_B = 10^{-8}$). When $||\Lambda||_2$ becomes lower than 1 we consider that we are close to a basin of attraction, and we give increased effect to regularization by setting $\theta_S = 10^{-4}$ and $\theta_B = 10^{-6}$. We use the numerical parameters of Table 2.2: in particular the FP steps are used at iterations 5 and 15.

Figure 2.7a represents the cost functional history during the optimization process. We observe first a rapid decrease of the cost functional during the gradient descent steps, which reaches a basin of attraction at iteration 9 with $\mathcal{J} = 0.23$ and $||\Lambda||_2 = 1.001 \times 10^{-2}$. Figure 2.7c shows the control provided by the algorithm for this iteration. The time evolution of the road densities with this control is represented in Figure 2.9a. The control performs rather well regarding the final cost functional and the final road densities on the central line. However, as might be expected, this remains rather unsatisfactory given the relatively high number of active controls and



(a) Road network and route to empty in red.



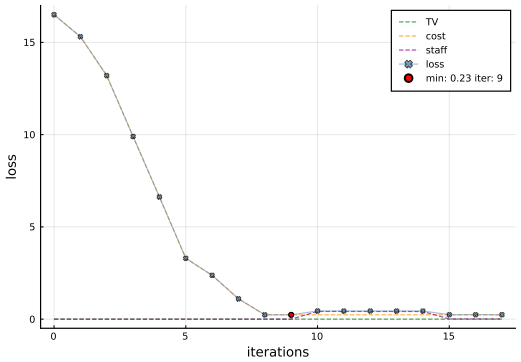
(b) Density at final time without control.

Figure 2.6: (Three lanes network) Left: Three lanes network configuration. Road 1 is incoming and road 11 is outgoing and all the others are directed from left to right. The route to empty $(4, 6, 7, 8, 9)$ is in red. Right: Numerical simulations without control at time $T = 30$.

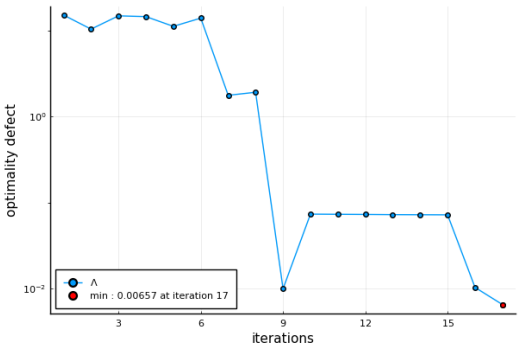
the redundancies. For instance, it should be useless to block the entrance of roads 18, 20 and 22 if roads 19, 21 and 23 are controlled.

We then observe a slight increasing of the loss $\mathcal{J}$ when the regularization coefficients θ_S and θ_B are increased, mainly due to the high values of $S(\mathbf{u})$ and $B_\nu(\mathbf{u})$. After the fixed-point step at iteration 15, all the coefficients of the loss are decreasing and reach the convergence tolerance by the 17-th iteration, as shown in Figure 2.7b. A zoom on this phenomena is given Figure 2.8.

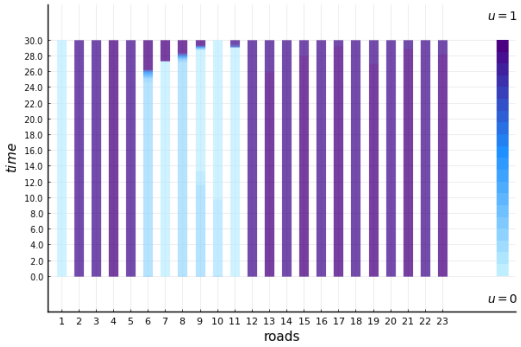
The cost $C_T(\mathbf{u})$ remains equal to 0.23 whereas the FP step at iteration 15 clearly reduced the staffing constraint $S(\mathbf{u})$ by more than an order of magnitude, at the cost of a slight increase in the total variation. Note that the GD steps were stationnary and that only the FP step led to convergence.



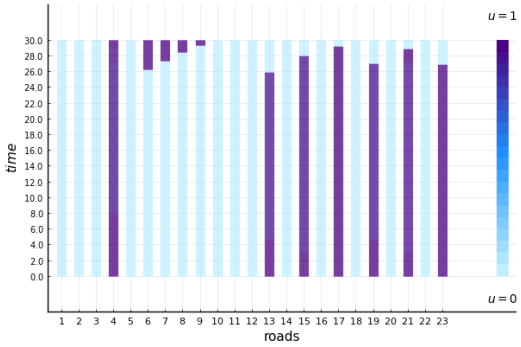
(a) Cost functional.



(b) $||\Lambda||_2$ history.

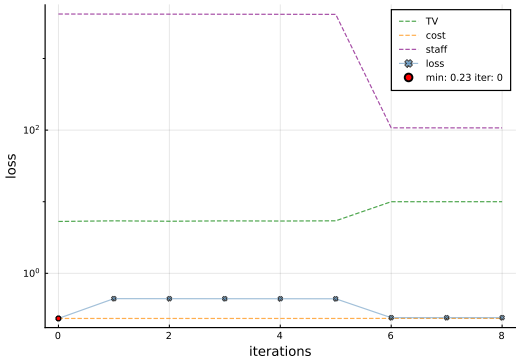


(c) Control (low constraints).

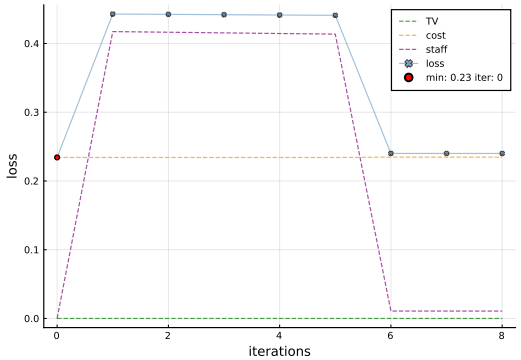


(d) Control (increased constraints).

Figure 2.7: (Three lanes network with scheduled constraints) Controls obtained with the GDFP algorithm.

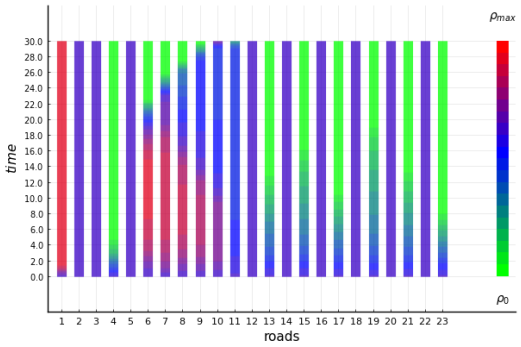


(a) Without weights.

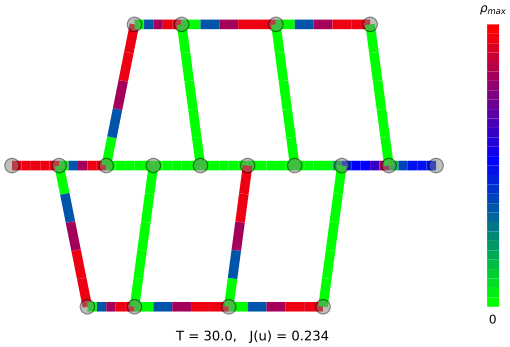


(b) With weights.

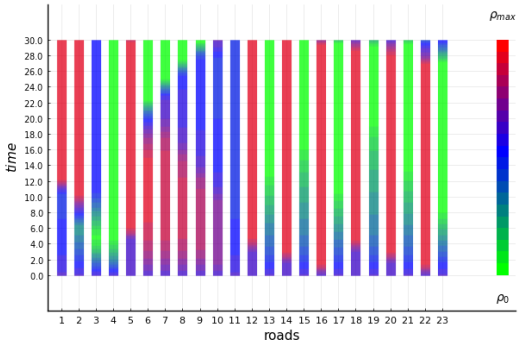
Figure 2.8: Cost functional between iterations 9 and 17.



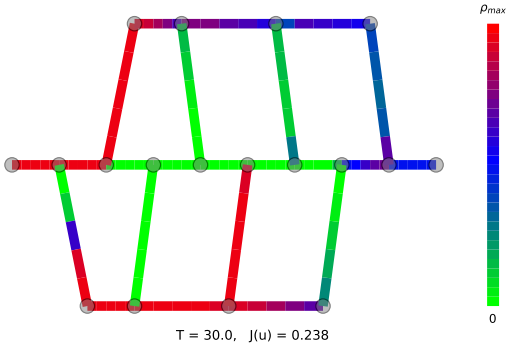
(a) Mean density (low constraints).



(b) Final time snapshot (low constraints).



(c) Mean density (increased constraints).



(d) Final time snapshot (increased constraints).

Figure 2.9: (Three lanes network with scheduled constraints) Densities obtained with the GDFP algorithm.

The penalization algorithm has clearly solved the two issues: the number of active controls is now less than 7, which is still higher than 5 but far preferable than the earlier 16 in a approached control point of view, and there is no time oscillations of the controls even if it would have been a mathematically correct way to reduce the staffing constraint.

Symbol	Name	Value
N_{max}	maximum number of simultaneous blockages allowed	5
θ_S^0	initial coefficient for the staffing constraint	10^{-8}
θ_S	scheduled coefficient for the staffing constraint	10^{-4}
θ_B^0	initial coefficient for the BV constraint	10^{-8}
θ_B	scheduled coefficient for the BV constraint	10^{-6}
ν	smoothed absolute value coefficient	10^{-10}

Table 2.3: Numerical parameters for the full model.

2.6 Conclusion

This article has therefore introduced a new network traffic control problem where the control acts not only on the flux constraints at the entrance of the road but also on the statistical behaviour matrices at the junctions. The theoretical analysis of the optimal control problem led to an original numerical method combining the gradient descent method and a fixed-point strategy. The numerical results obtained on the traffic circle and on the

three-lane network have shown that the method is relatively effective in providing relevant controls. It is also possible to impose constraints on the number of active controls.

This work can be pursued in several directions. Firstly, it would be interesting to show theoretically that the optimal controls are indeed bang-bang, as suggested by the numerical results. This could provide new improvements of the optimization algorithm. Secondly, the indicator and projection functions make our algorithms very sensitive to threshold or scaling effects. Further analysis could be of great help in determining hyperparameters such as θ or κ generically, since they are set by hand for the time being. Finally, the problem has been tackled here using a discretize (in space) then optimize strategy. Another possibility would have been to first optimize and then discretize. Such strategy may result in another kind of algorithms.

Obviously, it would be interesting to test the algorithm on larger road networks. For this, it might be interesting to couple this method with interesting initialisation methods where an initial control would be obtained with a macroscopic graph. Such an approach has, for example, been developed in a different context in epidemiology [27].

Appendix

Expression of the function ϕ^{LP}

We provide hereafter the explicit expression of ϕ^{LP} for junctions with at most 2 ingoing and 2 outgoing roads. We will write $(\boldsymbol{\gamma}^R, \boldsymbol{\gamma}^L) = \phi^{LP}(\boldsymbol{\rho}, \mathbf{u})$.

One ingoing and one outgoing roads ($n = m = 1$). In that case, the solution of the LP problem (2.3) is given by

$$\gamma_1^R = \gamma_2^L = \min\left(\gamma_1^{R,\max}, (1-u_2)\gamma_2^{L,\max}\right).$$

One ingoing and two outgoing roads ($n = 1, m = 2$). Setting $\alpha = \alpha_{21}$, we obtain

$$\gamma_1^R = \min\left(\gamma_1^{R,\max}, \min\left((1-u_2)\frac{\gamma_2^{L,\max}}{\alpha}, (1-u_3)\frac{\gamma_3^{L,\max}}{1-\alpha}\right)\right),$$
$$\gamma_2^L = \alpha\gamma_1^R,\ \gamma_3^L = (1-\alpha)\gamma_1^R.$$

However, this formulation raises a modeling problem. Indeed, this solution does not distinguish between a blocked road ($\gamma_j^{R,\max} = 0$) and a controlled road ($1 - u_j = 0$). Indeed, the first case corresponds to a congestion involving cars on both sides of the intersection, preventing vehicles from entering the second -potentially empty- road, while in the second case, drivers will not try to enter the controlled road and we revert to the 1×1 configuration with the remaining roads.

To capture this difference in driver behavior, we will adjust the parameter α with respect to the control $\mathbf{u}$. Therefore, α will be considered as a function of (u_2, u_3). More precisely, we introduce a function denoted P of the variable $x = u_2 - u_3$, taking the constant value $\bar{\alpha} \in (0, 1)$ at $x = 0$ and yielding the relevant 1×1 case whenever one road is fully controlled. For the sake of simplicity, we look for a function $(u_2, u_3) \mapsto P(u_2 - u_3)$,

where P is a polynomial of degree at most 2, that satisfies

$$P(-1) = \alpha(0,1) = 1, \quad P(0) = \alpha(u,u) = \bar{\alpha}, \quad P(1) = \alpha(1,0) = 0.$$

The standard Lagrange interpolation formula yields

$$P(x) = \frac{x(x-1)}{2} + \bar{\alpha}(1-x^2). \tag{2.26}$$

However, due to the division by α in the expression of γ_1^R, the case $\alpha(u_2 = 1, u_3 = 0) = 1 - u_2 = 0$ is degenerate. To overcome this definition problem, we introduce a small perturbation parameter $\varepsilon > 0$ and a modification of the polynomial P denoted P_ε, such that $\alpha_\varepsilon := P_\varepsilon(u_2 - u_3)$ and

$$P_\varepsilon(-1) = 1 - \varepsilon^2, \quad P_\varepsilon(0) = \alpha, \quad P_\varepsilon(1) = \varepsilon^2.$$

The same reasoning as above yields $P_\varepsilon(x) = P(x) + \varepsilon^2 x$. Since the interpolation procedure does not guarantee that the range of the function be contained in $[\varepsilon^2, 1-\varepsilon^2]$, we compose the obtained expression with a projection onto the set of admissible values, to get at the end

$$\alpha_\varepsilon(u_2, u_3) := \mathrm{proj}_{[\varepsilon^2, 1-\varepsilon^2]}\left(P_\varepsilon(u_2 - u_3)\right). \tag{2.27}$$

Similarly, we also replace the expression $(1-u_j)$ by $(1-u_j+\varepsilon)$ to avoid the case where $(1-u_2) = \alpha(u_2, u_3) = 0$. We finally obtain the following regularized solution at 1×2 junctions

$$\gamma_1^R = \min\left(\gamma_1^{R,\max}, \min\left((1-u_2+\varepsilon)\frac{\gamma_2^{L,\max}}{\alpha_\varepsilon(u_2,u_3)}, (1-u_3+\varepsilon)\frac{\gamma_3^{L,\max}}{1-\alpha_\varepsilon(u_2,u_3)}\right)\right),$$
$$\gamma_2^L = \alpha_\varepsilon \gamma_1^R,\ \gamma_3^L = (1-\alpha_\varepsilon)\gamma_1^R.$$

Two ingoing and one outgoing roads ($n = 2$, $m = 1$). The ingoing roads are indexed by $1, 2$ and the outgoing one is indexed by 3. In any case, we have $\gamma_3^L = \gamma_1^R + \gamma_2^R$, and γ_1^R, γ_2^R are computed as follows:

- we set $\gamma_{3,\mathbf{u}}^{L,\max} := (1-u_3)\,\gamma_3^{L,\max}$;
- if $\gamma_1^{R,\max} + \gamma_2^{R,\max} \le \gamma_{3,u}^{L,\max}$, then $\gamma_1^R = \gamma_1^{R,\max}$ and $\gamma_2^R = \gamma_2^{R,\max}$;
- else,
  - if $\gamma_1^{R,\max} \ge q\,\gamma_3^{L,\max}$ and $\gamma_2^{R,\max} \ge (1-q)\gamma_{3,u}^{L,\max}$, then $\gamma_1^R = q\,\gamma_{3,u}^{L,\max}$ and $\gamma_2^R = (1-q)\,\gamma_{3,u}^{L,\max}$.
  - if $\gamma_1^{R,\max} < q\gamma_{3,u}^{L,\max}$ and $\gamma_2^{R,\max} \ge (1-q)\gamma_{3,u}^{L,\max}$, then $\gamma_1^R = \gamma_1^{R,\max}$ and $\gamma_2^R = \gamma_3^{L,\max} - \gamma_1^{R,\max}$.
  - if $\gamma_1^{R,\max} \ge q\gamma_{3,u}^{L,\max}$ and $\gamma_2^{R,\max} < (1-q)\gamma_{3,u}^{L,\max}$, then $\gamma_1^R = \gamma_{3,u}^{L,\max} - \gamma_2^{R,\max}$ and $\gamma_2^R = \gamma_2^{R,\max}$.

Two ingoing and two outgoing roads ($n = m = 2$). The ingoing roads are indexed by $1, 2$ and the outgoing ones are indexed by $3, 4$. We have

$$\begin{pmatrix}\gamma_3 & \gamma_4\end{pmatrix}^\top = A(\mathbf{u})\begin{pmatrix}\gamma_1 & \gamma_2\end{pmatrix}^\top \quad \text{with} \quad A(\mathbf{u}) = \begin{pmatrix}\alpha(\mathbf{u}) & \beta(\mathbf{u}) \\ 1-\alpha(\mathbf{u}) & 1-\beta(\mathbf{u})\end{pmatrix},$$

where, given two real number $\bar{\alpha}$, $\bar{\beta}$ in $(0,1)$, the coefficients are defined in the same way as in the case 1×2, as

$$\alpha_\varepsilon(u) = \mathrm{proj}_{[\varepsilon^2, 1-\varepsilon^2]}\left(P_\varepsilon^{\bar{\alpha}}(u_3 - u_4)\right), \quad \beta_\varepsilon(u) = \mathrm{proj}_{[\varepsilon^2, 1-\varepsilon^2]}\left(P_\varepsilon^{\bar{\beta}}(u_3 - u_4)\right).$$

Here, P_ε^ξ is defined by

$$P_\varepsilon^\xi(x) = \frac{x(x-1)}{2} + \xi(1-x^2) + \varepsilon^2 x, \quad \xi \in \{\bar{\alpha}, \bar{\beta}\}. \tag{2.28}$$

Let us set $\gamma_{i,\mathbf{u}}^{L,\max} := (1-u_i)\,\gamma_i^{L,\max}$, for $i = 3, 4$.

To solve the LP 2×2 problem, we first analyze the simplex standing for the polytope of constraints, represented on Figure 2.10: the constraints related on the incoming routes form the rectangle $\Omega_{in} := [0, \gamma_1^{R,\max}] \times [0, \gamma_2^{R,\max}]$ while those related on the outgoing roads correspond to the regions below the curves $\mathcal{C}_3$ and $\mathcal{C}_4$, respective graphs of the functions

$$\gamma_1 \mapsto \frac{(1-u_3)\gamma_3^{L,\max} - \alpha_{\boldsymbol{u}}\gamma_1}{\beta_{\boldsymbol{u}}} \quad \text{and} \quad \gamma_1 \mapsto \frac{(1-u_4)\gamma_4^{L,\max} - (1-\alpha_{\boldsymbol{u}})\gamma_1}{1-\beta_{\boldsymbol{u}}}.$$

A convexity argument, standard in linear optimization, yields that the exists (at least) a solution lying on a vertex of the set of constraints. To discuss on optimality of each vertex, we first compute $\mathbf{\Gamma_u}$, the intersection of $\mathcal{C}_3$ and $\mathcal{C}_4$.

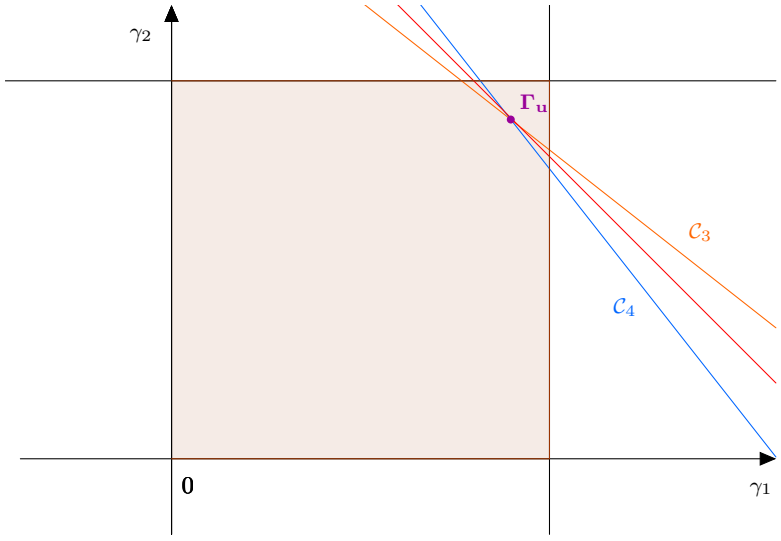


Figure 2.10: Polytope of constraints for the 2x2 LP. The constraints on the input flow are represented by the colored area.

The point $\mathbf{\Gamma_u} = (\Gamma_{1\mathbf{u}}, \Gamma_{2\mathbf{u}})$ is the intersection point of the lines with cartesian equations $\alpha_{\boldsymbol{u}}\gamma_1 + \beta_{\boldsymbol{u}}\gamma_2 = \gamma_{3,\mathbf{u}}^{\max}$ and $(1-\alpha_{\boldsymbol{u}})\gamma_1 + (1-\beta_{\boldsymbol{u}})\gamma_2 = \gamma_{4,\mathbf{u}}^{\max}$. We obtains

$$\begin{cases} \Gamma_{1\mathbf{u}} = \left((1-\beta_{\boldsymbol{u}})\gamma_{3,\mathbf{u}}^{\max} - \beta_{\boldsymbol{u}}\gamma_{4,\mathbf{u}}^{\max}\right)/\triangle_{\mathbf{u}}, \\ \Gamma_{2\mathbf{u}} = \left(-(1-\alpha_{\boldsymbol{u}})\gamma_{3,\mathbf{u}}^{\max} - \alpha_{\boldsymbol{u}}\gamma_{4,\mathbf{u}}^{\max}\right)/\triangle_{\mathbf{u}}, \end{cases} \tag{2.29}$$

with $\triangle_{\mathbf{u}} = \alpha_{\boldsymbol{u}}(1-\beta_{\boldsymbol{u}}) - \beta_{\boldsymbol{u}}(1-\alpha_{\boldsymbol{u}})$. A standard graphical easoning shows that the solution is realized at $\mathbf{\Gamma_u}$ if $\mathbf{\Gamma_u}$ belongs to Ω_{in}, as in Figure 2.10. Otherwise, we have to intersect the constraint of highest slope (in absolute value) with Ω_{in}. It is this disjunction that is performed in what follows.

- **First case:** $\Gamma_{1\mathbf{u}} \leq \gamma_1^{\max}$ and $\Gamma_{2\mathbf{u}} \leq \gamma_2^{\max}$;
  - if $\Gamma_{1\mathbf{u}} >= 0$ and $\Gamma_{2\mathbf{u}} < 0$, then $\gamma_2 = 0$;
    - if $\alpha_{\boldsymbol{u}} < \beta_{\boldsymbol{u}}$, then $\gamma_1 = \gamma_{3,\mathbf{u}}^{\max}/\alpha_{\boldsymbol{u}}$;
    - else, $\gamma_1 = \gamma_{4,\mathbf{u}}^{\max}/(1-\alpha_{\boldsymbol{u}})$.
  - if $\Gamma_{1\mathbf{u}} < 0$ **and** $\Gamma_{2\mathbf{u}} \geq 0$, then $\gamma_1 = 0$;
    - if $\alpha_{\boldsymbol{u}} < \beta_{\boldsymbol{u}}$, then $\gamma_2 = \gamma_{3\mathbf{u}}^{\max}/\beta_{\boldsymbol{u}}$;
    - else, $\gamma_2 = \gamma_{4\mathbf{u}}^{\max}/(1-\beta_{\boldsymbol{u}})$.
  - if $\Gamma_{1\mathbf{u}} < 0$ **and** $\Gamma_{2\mathbf{u}} < 0$, then $\gamma_1 = \gamma_2 = 0$;
  - else, $(\gamma_1, \gamma_2) = (\Gamma_{1\mathbf{u}}, \Gamma_{2\mathbf{u}})$.
- **Second case:** $\Gamma_{1\mathbf{u}} > \gamma_1^{\max}$ and $\Gamma_{2\mathbf{u}} > \gamma_2^{\max}$. Then, $(\gamma_1, \gamma_2) = (\gamma_1^{\max}, \gamma_2^{\max})$.
- **Third case:** $\Gamma_{1\mathbf{u}} > \gamma_1^{\max}$ and $\Gamma_{2\mathbf{u}} \leq \gamma_2^{\max}$;
  - if $\alpha_{\boldsymbol{u}} < \beta_{\boldsymbol{u}}$, then

$$\gamma_2 = \mathrm{proj}_{[0,\gamma_2^{\max}]}\left(\frac{\gamma_{3\mathbf{u}}^{\max} - \alpha_{\boldsymbol{u}}\gamma_1^{\max}}{\beta_{\boldsymbol{u}}}\right), \quad \gamma_1 = \begin{cases} \gamma_3^{\max}/\alpha_{\boldsymbol{u}} & \text{if } \gamma_2 = 0 \\ \gamma_1^{\max} & \text{if } \gamma_2 \in (0, \gamma_2^{\max}]; \end{cases}$$

  - else,

$$\gamma_2 = \mathrm{proj}_{[0,\gamma_2^{\max}]}\left(\frac{\gamma_{4\mathbf{u}}^{\max} - (1-\alpha_{\boldsymbol{u}})\gamma_1^{\max}}{1-\beta_{\boldsymbol{u}}}\right), \quad \gamma_1 = \begin{cases} \gamma_4^{\max}/(1-\alpha_{\boldsymbol{u}}) & \text{if } \gamma_2 = 0 \\ \gamma_1^{\max} & \text{if } \gamma_2 \in (0, \gamma_2^{\max}]; \end{cases}$$

- **Fourth case:** $\Gamma_{1\mathbf{u}} \leq \gamma_1^{\max}$ and $\Gamma_{2\mathbf{u}} > \gamma_2^{\max}$;
  - if $\alpha_{\boldsymbol{u}} > \beta_{\boldsymbol{u}}$, then

$$\gamma_1 = \mathrm{proj}_{[0,\gamma_1^{\max}]}\left(\frac{\gamma_{3\mathbf{u}}^{\max} - \beta_{\boldsymbol{u}}\gamma_1^{\max}}{\alpha_{\boldsymbol{u}}}\right), \quad \gamma_2 = \begin{cases} \gamma_3^{\max}/\beta_{\boldsymbol{u}} \text{if } \gamma_1 = 0 \\ \gamma_2^{\max} \qquad\qquad\qquad \text{if } \gamma_1 \in (0, \gamma_1^{\max}]; \end{cases}$$

  - else,

$$\gamma_1 = \mathrm{proj}_{[0,\gamma_1^{\max}]}\left(\frac{\gamma_{4\mathbf{u}}^{\max} - (1-\beta_{\boldsymbol{u}})\gamma_1^{\max}}{1-\alpha_{\boldsymbol{u}}}\right), \quad \gamma_2 = \begin{cases} \gamma_4^{\max}/(1-\beta_{\boldsymbol{u}}) & \text{if } \gamma_1 = 0 \\ \gamma_2^{\max} & \text{if } \gamma_1 \in (0, \gamma_1^{\max}]. \end{cases}$$

Regular approximation of ϕ^{LP}. In the following, in order to use differentiable optimization techniques, we will replace the function ϕ^{LP}, which is only Lipschitz, by an approximation C^1, replacing all operations consisting in taking the minimum or the maximum of two quantities by regular approximations:

$$\min\{x,y\} \leftarrow \frac{x+y-\sqrt{(x+y)^2+\eta^2}}{2} \quad \text{and} \quad \max\{x,y\} \leftarrow \frac{x+y+\sqrt{(x+y)^2+\eta^2}}{2}$$

for a given $\eta > 0$.

Remark 5 (Some remarks on the LP problem)**.** *It is notable that In [78], a close LP problem has been solved. However, we found that a case corresponding to a particular constraint configuration was forgotten in their analysis. This error could lead to erroneous vertices provifding negative values to some flows. The configuration that was not taken into account in this publication is illustrated on Figure 2.11, for parameter choices from Table 2.4.*

Parameter	$\gamma_1^{\max}$	$\gamma_2^{\max}$	$\gamma_3^{\max}$	$\gamma_4^{\max}$	u_3	u_4	α	β
Value	0.25	0.25	0.2244	0.2244	0	0.96	0.45	0.50

Table 2.4: Parameters for the 2x2 LP problem in Figure 2.11.

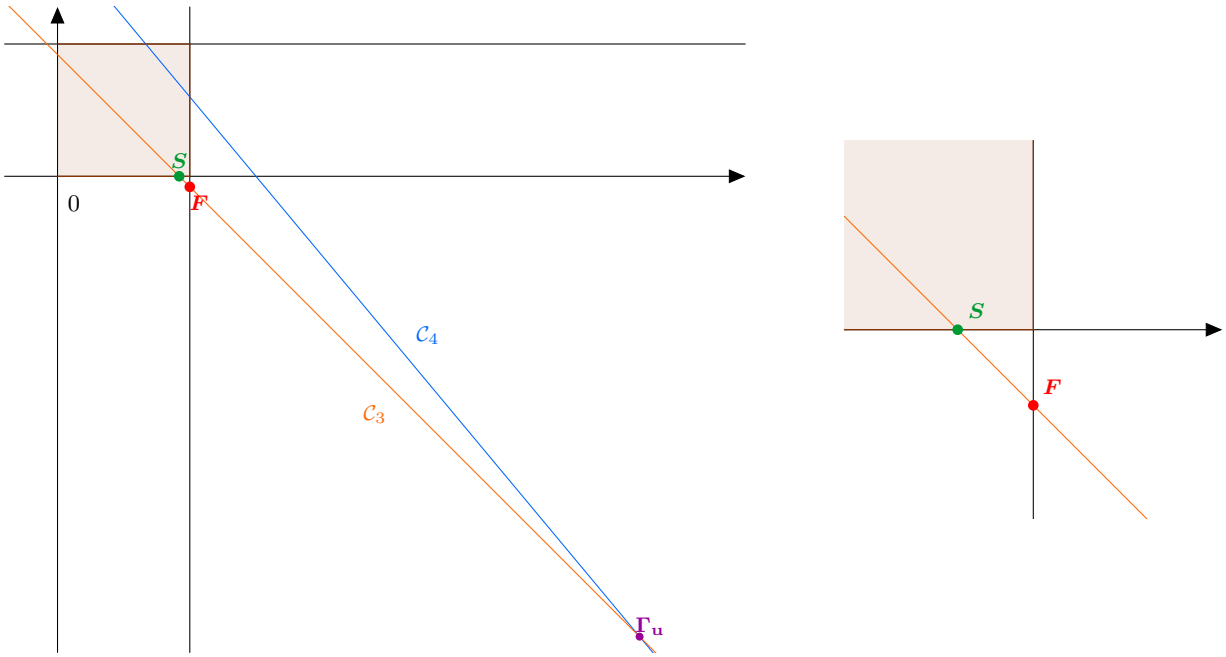


Figure 2.11: Graphical resolution of the 2×2 LP problem. polytope of constraint (left) and a zoom (right). $\boldsymbol{F}$ is the erroneous solution and $\boldsymbol{S}$ is the correct one.

Chapter 3

A Julia code for optimal control of a road network

This chapter aims at valorizing and making exploitable the code used for the numerical simulations of the our article presented in chapter 2, that can be found at `https://github.com/mickaelbestard/TRoN.jl`, as well as retracing some programming notions learned during this thesis. To do so, we will first present the programming interface allowing to reproduce identically the results presented previously, as well as to create new test cases from the existing graphs *(front-end: familiarization with the interface and customization of existing function calls)*. In a second more technical part, we will go into more detail on the design of the interface itself in order to give the interested reader the possibility to tailor the code to his needs, with for example the customization of the cost function, the optimization algorithm or the graph representing the road network. The final part will deal with the limitations and the ways to improve the code in a perspective of scaling up on large problems, like for example on a whole city. *(back-end: reappropriation of the code while being aware of some key aspects of the Julia language)*.

1 Why Julia? Overview of the code

An efficient implementation of the algorithms presented above implies two objectives that may be contradictory at first sight. First, we need a language that is sufficiently advanced to allow us to easily build our model, such as the graph structure or the automatic gradient calculations. On the other hand, the complexity of our model requires high numerical performance, especially on larger graphs. While the usual workflow is often to first write a small proof of concept in Python or Matlab before rewriting and optimizing it in a compiled language such as `C/C++/Fortran`, we chose to develop and optimize (in the numerical sense) in the same language. Julia seems indeed well suited to this purpose.

However, if one of the main ambitions of Julia is to offer the syntactic simplicity of Python with the performance of C, it can be very helpful to understand the internal mechanisms of Julia, and in particular the management of memory allocations, in order to take full advantage of this language and manage to write code that is efficient, generic and easily readable. This is why we have grouped most of the code complexity in a simplified user interface that we will explain in the first sections.

Files organization: We use a common project organization, with a separation of folders into ‘sources‘, ‘tests-cases‘ and ‘unit-tests‘. Packages are managed via a ‘Pkg‘ environment which allows to create a local module, which we call TRON (TRaffic On Networks). This also avoids compatibility problems between different possible versions of the libraries, making the project easier to maintain.

The informations about the problem are distributed along different data structures[1]:

- `Junction`, containing the local data of the nodes such as the adjacent routes, static and dynamic distribution matrices, priorities parameters...
- `Network`, which centralizes all data relating to the network, i.e. the directed graph with local metadata such as maximum local speed and density, the set of junctions, the LWR model flow and its discretization to a numerical flow, ...
- `Mesh`, the spatial discretization of a road, i.e. the cells with their area, the lengths of each road[2], ...
- `Model`, that unifies all previously introduced physical quantities and stores the computations related to the direct problem as well as the jacobians of f^{FV} and ϕ^{LP}.
- The structure `Optim` connects the road graph with our optimizers in order to perform the optimal control algorithm.
- Finally, `Diagnostics` is a placeholder that concisely gathers all the data to be saved/plotted.

which interact with the following keys external libraries:

- `Graph.jl` which implements for instance the directed graph structure with useful functions to easily manipulate it (vertices(graph::SimpleDiGraph), edges(graph::SimpleDiGraph), source or destination of an edge, ...)
- `ForwardDiff.jl` which is used for automatic differentiation in the structure `Model`.

The arborescence is summarized in the UML diagram Figure 3.1. The convention used is that custom structures are in bold, existing libraries are in italics, and the "+" and "-" signs denote public and private fields, respectively.

2 Presentation of the interface

This part serves as a general presentation of the code and gives the necessary indications for its use as a standalone tool.

You must have cloned the following git directory

`https://github.com/mickaelbestard/TRoN.jl`

and launched the Tron module (cf readme.md).

Pre-configured test cases are located in the folder `Tron/example`, containing the subfolders:

[1] Julia does not use objects, but structures. We present as "methods" functions that call and modify or use the attributes of such structures.

[2] In this version of the code all roads are assumed to have the same lengths, but it is possible to generalize the implementation readily.

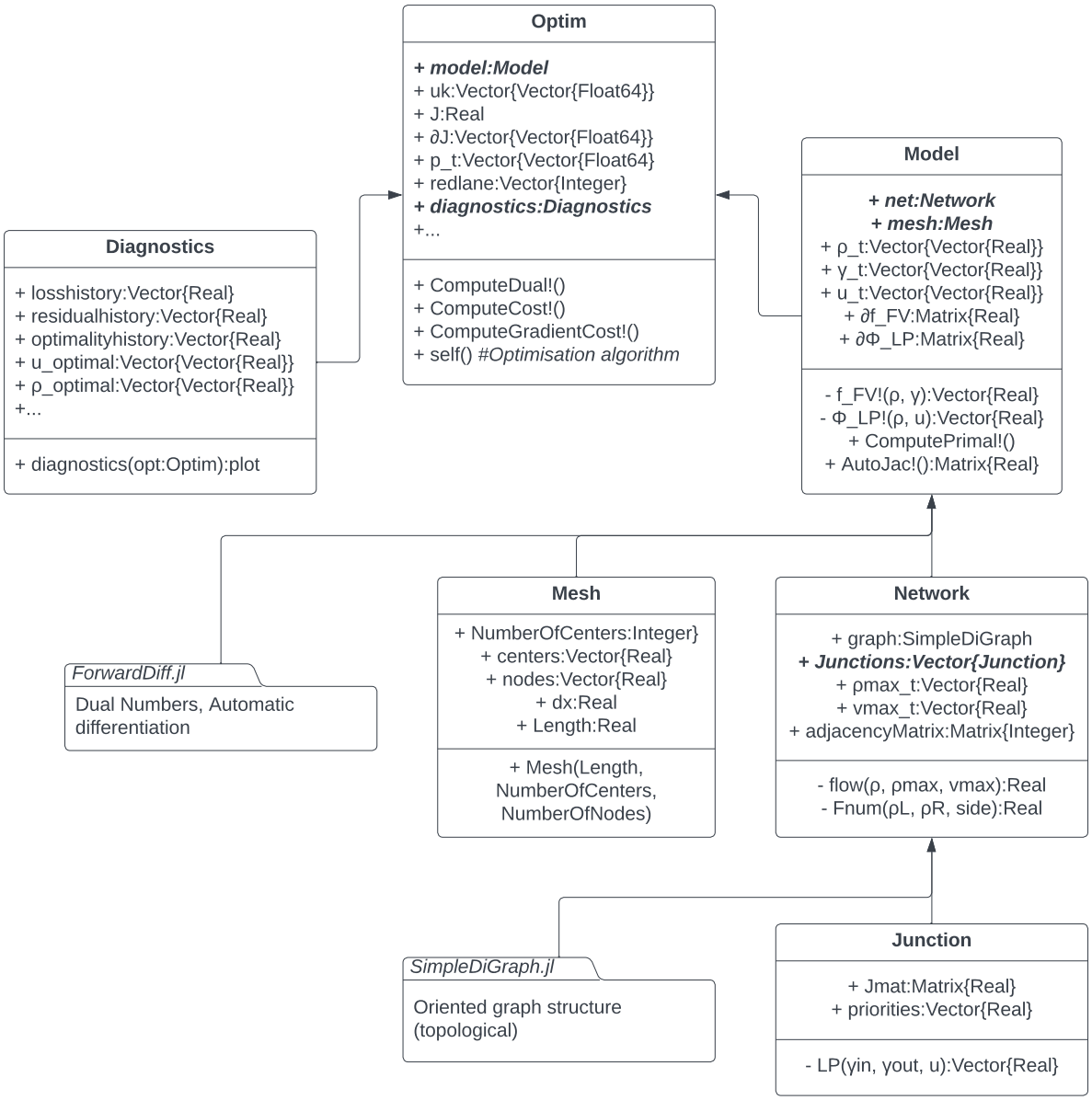


Figure 3.1: UML diagram of main code structures.

- `junctions: 1x1, 1x2, 2x1, 2x2`
- `circle`
- `threeways`

As an example, let us consider the test case of a junction with one input and one output, located in the folder `junctions/1x1/`. This folder contains the file `Test_1x1.jl`, which we import via

```
1 include("Test_1x1.jl")
```

In order to trigger the compilation, it is usual to run the code a first time with small parameters. This can be done here by using the default ones with the following function call[3]:

```
1 test_Optimizer();
```

Once this is done, we launch the proper optimization loop and store the result afterwards:

```
1 opt = test_Optimizer(final_time=6, Nc=50, algo="GD")
```

The possible values of the parameter `algo`, corresponding to the different choices of optimization algorithms in chapter 2 are:

- `"GD"` : projected gradient descent
- `"FP"` : fixed point algorithm
- `"GDFP"` : hybridization of GD and FP algorithms

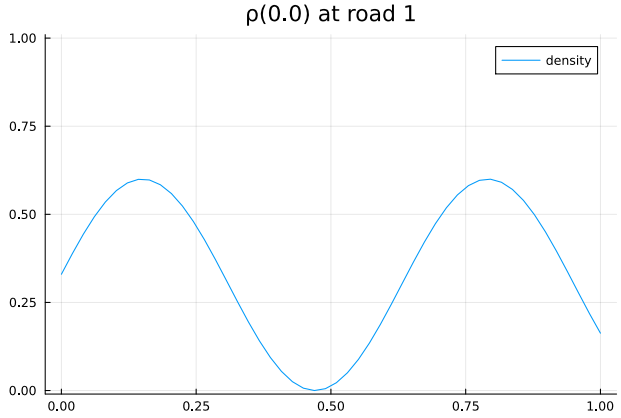
Notice that other parameters such as penalized constraints and convergence accuracy can also be prescribed, but the default values are those of the paper and can be left unchanged for now.

We then retrieve the results via

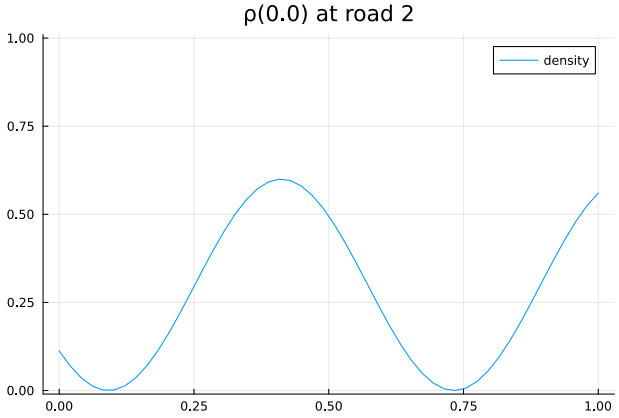
```
1 diagnostics(opt, test_case="junction")
```

which produces the convergence graphs as well as the optimal snapshots and places them in a sub-folder `diagnostics` of the current folder. The `test_case` parameter is optional, and is used to obtain a heatmap instead of a standard 1D graphs, see Figure 3.2. The value `"junction"` is common to the four test cases of the `junction` folder, and can be replaced respectively by `"circle"` or `"threeways"` in the corresponding test cases.

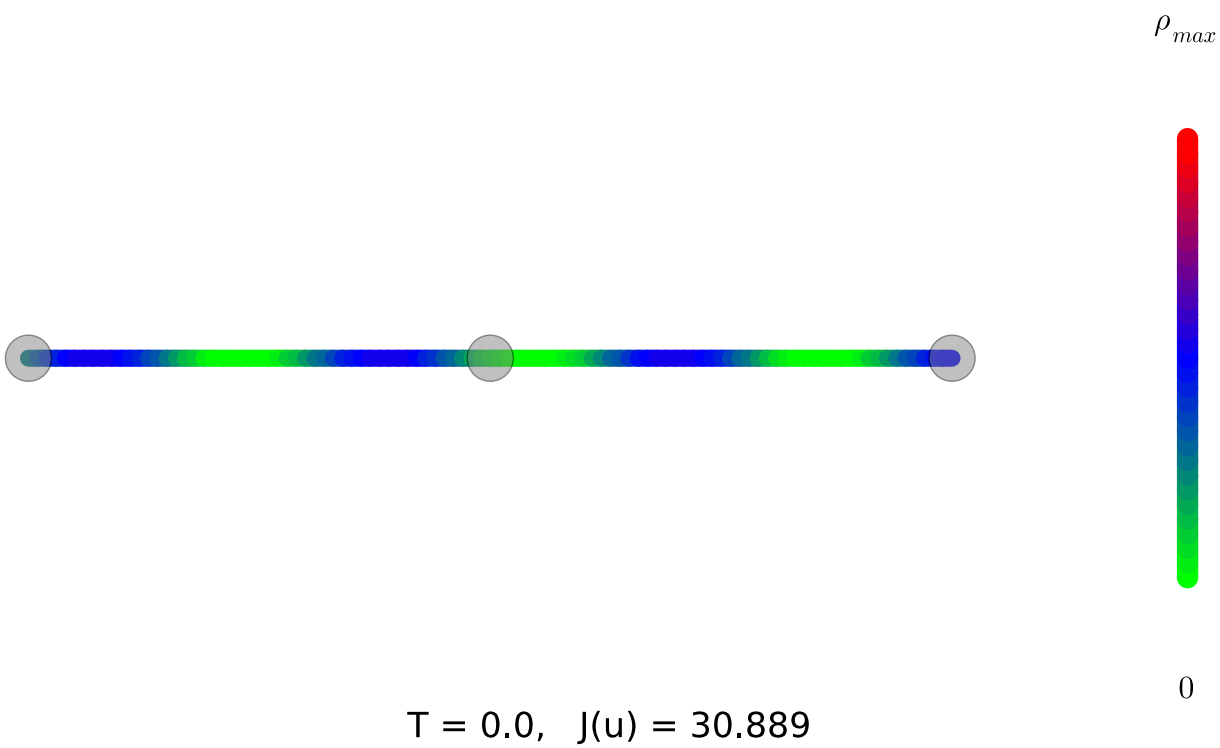
[3]The semicolon is used to prevent the returned object from being displayed, and is not necessary as it would be in C, for example.



(a) Density at initial time on road 1 (local coordinates)



(b) Density at initial time on road 2 (local coordinates)



(c) Density at initial time on the 1x1 network.

Figure 3.2: Standard 1D graphics and heatmap display of the 1x1 junction. Plotted density: $0.3(\sin(5x)+1)$, for $0 \leq x \leq 2$ and $\Delta x = 1/50$.

3 Editing an existing test case

Let's now dig a little deeper into the code to modify the test parameters. To do this, let's first unpack the function `test_Optimizer`.

```
1  function test_Optimizer(;final_time=0.3, Nc=5, Nmax=2, coef_pen=0, coef_reg=0,
2      itermax=3, CV_tol=1e-2, algo="GD", lane=Bool.([1,1]), u_init=0.0,
3      initstate=Constant(value=0.66), fp_tol=0.0, fp_trigger=10, delay_factor=1.0,
4      ismaskcontrol=false, network = graph_1x1(1.0), lr_init=1e-1, decay=1e-1
5      )
6
7      model = Model(network, Nc, Tf=final_time)
8      init(model)
9
10     optim = Optim(model, Nmax, coef_pen, coef_reg)
11     set_scheduler(optim, lr_init=lr_init, decay=decay)
12     set_redlane!(optim, lane)
13     optim.fp_tol = fp_tol
14     optim.trigger = fp_trigger
15     optim.delay_factor = delay_factor
16     optim.ismaskcontrol = ismaskcontrol
17     [u .= u_init for u in optim.ukm]
18     [u .= u_init for u in optim.uk]
19     [u .= u_init for u in optim.ukp]
20
21     optim(itermax=itermax, CV_tol=CV_tol, algo=algo, initstate=initstate)
22
23     return optim
24 end
```

Line by line:

lines 1-4: Function call parameters, cf Table 3.1.

lines 7-8: A controlled model is initialized from the network. The latter can be chosen among the other pre-configured ones (1x2, circle, ...) or modified via the parameters (given here with their default values) `rmax_t=[1., 1.]`, `vmax_t=[1., 1.]`, `density_at_leaves_t=[0.66, 0.]`. These parameters are vectors whose i-th coordinate applies to the i-th route.

line 10: Creation of a structure containing the data of the control problem

lines 11-19: Assigning parameters in the optimization structure

line 21: The structure is also callable as a function. Launching the optimization loop.

The structure `optim` is returned to the user and contains all the data of the problem.

symbol	meaning	type
final_time	Final time (dimensionless) for the direct problem	Float
Nc	Number of cells per route	Int
Nmax	Maximum number of simultaneous controls allowed	Int
coef_pen	Penalization coefficient θ_S of the constraint $\sum_i u_i(t) \leq N_{\max}$	Float
coef_reg	Penalization coefficient θ_B of the BV regularization	Float
itermax	Maximum number of iterations for the optimization algorithm	Int
lane	"One-hot vector" indicating the routes to be evacuated	Vector{Bool}
u_init	Control initial value	Float
initialstate	Initial condition of the direct problem	**InitFunction** *(custom)*
fp_tol	Tolerance on the sign of the gradient for the fixed point algorithm	Float
fp_trigger	FP triggering delay in GDFP	Int
delay_factor	spacing of fp_trigger in GDFP	Float
CV_tol	Convergence tolerance for the optimization algo	Float
ismaskcontrol	Indicates if we restraint some roads to be controlled	Bool
network	Road network to optimize	**Network** *(custom)*
lr_init	Initial learning rate (can be scheduled)	Float
decay	Decay in the lr scheduler	Float

Table 3.1: Call parameters of the function `test_Optimizer`.

4 Customize the code

The purpose of this section is to show what to modify in the code to solve other similar problems.

Create a graph: Let's first study how to implement its own graph through the example of the design of the 1x1 graph, whose code is given by

```
function graph1x1(;rmax_t=[1.,1.], vmax_t=[1.,1.], density_at_leaves_t=[0.66,0])

    Nr = 2
    Nv = 3
    Nl = 2

    Net = Network(Nr, Nv, Nl, rmax_t, vmax_t)

    # graph topology
    add_edge!(Net.graph, 1,2)
    add_edge!(Net.graph, 2,3)

    # placement of "leafs" nodes and corresponding density values
    # Graph is: 1 (enter node) --> 2 (junction) --> 3 (exit node)
    set_list_leaves(Net, [1,3])
    set_ρ_leaves_t(Net, density_at_leaves_t)

    init_Network(Net)
```

```
19
20      # setting repartition matrices
21      J1x1 = ones((1,1))
22      set_Jmat(Net.Junctions[1], J1x1)
23      set_Jmat(Net.Junctions[2], J1x1)
24      set_Jmat(Net.Junctions[3], J1x1)
25
26      return Net
27  end
```

Lines 3 to 7 are used to create the structure `Network` which contains the macroscopic LWR model as well as what could be called a topological directed graph, the nodes having no coordinates.

The graph structure is coded via the library `Graph.jl` which allows, once a graph is initialized, to link node i to node j via `add_edge!(graph, i, j)`.

We then indicate the nodes of the network for which we will have to impose conditions on the edges, i.e. the leaves of the graph. Note that for now this code assumes constant boundary conditions at leaves, it has to be modified to make them time-dependant.

Lines 21 to 24 are used to create and assign the default distribution matrix at each junction. It is given with constant values but will later depend on the value of the control, see chapter 2 for more explanations on the mathematical model.

It is important to note that the size of each route is assumed to be constant equal to 1 throughout the code, and the same is true for the number of cells per route.

Edit optimizer: If we now want to change the way the descent direction is calculated, we need to implement a function with the following signature:

```
1  function my_descent(opt::Optim)
2      # some julia code
3  end
```

The Optim structure `opt::Optim` notably contains the following data:

- control at the current iteration Optim.uk = $\mathbf{u}^k$ and at the previous iteration Optim.ukm = $\mathbf{u}^{k-1}$
- the gradient of the cost function to be minimized, at the current iteration: Optim.dJ = $\nabla J(\mathbf{u}^k)$
- tolerance threshold for gradient cancellation Optim.κ = κ (in the fixed-point algo, a gradient coordinate is considered as null if its absolute value is below this threshold)

The only constraint that this function must respect is to return a descent direction $\Delta\mathbf{u}^k$, so that the iteration of the control loop is always written in the form :

$$\mathbf{u}^{k+1} \leftarrow \mathrm{proj}(\mathbf{u}^k - \Delta\mathbf{u}^k),$$

where proj is the projection of the coordinates between 0 and 1. This translates in particular for the fixed-point algorithm by the following expression:

```
function fixed(opt::Optim)
    return opt.uk .-
        (1. * map.(x -> x < -opt.κ, opt.dJ)
        + myprod(opt.uk, 1. * map.(x -> -opt.κ <= x <= opt.κ, opt.dJ) )
        )
end
```

where the returned value is $\Delta \mathbf{u}^k := \mathbf{u}^k - FP = \mathbf{u}^k - (\mathbb{1}_{\nabla J(\mathbf{u}^k) < -\kappa} + \mathbf{u}^k \mathbb{1}_{|\nabla J(\mathbf{u}^k)| \leq \kappa})$ so that we have

$$\mathbf{u}^{k+1} \leftarrow \text{proj}(\mathbf{u}^k - \Delta \mathbf{u}^k) = \mathbf{u}^k - (\mathbf{u}^k - FP) = FP,$$

with all the coordinates of FP being in $[0, 1]$ by construction.

Minimize another functional: Our optimal control software can easily address other problems by modifying the cost functional. We may want to minimize the average speed on some roads, the pollution, etc...

The function to be called to calculate the cost function is noted `ComputeCost!`, and can be very generic under the only conditions that it :

- takes `opt::Optim` as input
- modifies the attribute `opt.J`
- returns `opt.J`, given the current implementation of the linear search algorithm (see line 6):

```
    ### linesearch
    iter=0
    while J1 >= J0 && iter < maxiter
        descent_step *= decay_coef
        ukp_guess = proj.(opt.uk .- descent_step*opt.dJ)
        J1 = ComputeCost!(opt, u=ukp_guess)
        iter += 1
    end
```

As an example, here is the implementation of the cost function used in the article[4] :

```
function ComputeCost!(opt::Optim; u=opt.model.u_t,
                      initialfunc=Constant(value=0.66))

    θs, θb = opt.regularizations
    set_control!(opt.model, u)
    ComputePrimal!(opt.model, initialfunc)
    Nmax_deviation = 0.
    TVU = 0.
    if opt.ispenalized
        for n in eachindex(u)
            Nmax_deviation += max(0, sum(u[n]) - opt.Nmax)^2 * opt.model.Δt_t[n]
```

[4] IMPORTANT: in Julia, the last computed value is automatically returned by the function, hence the absence here of the **`return`** instruction.

```
12          end
13      end
14      if opt.isregularized
15          TVU = TotalVariations(u)
16      end
17      ρT_sum = dot(opt.redlane, opt.model.ρ_t[end])
18
19      opt.J = ρT_sum + θs * Nmax_deviation + θb * TVU
20  end
```

5 Limitations of the code and perspectives

Although this code is somewhat efficient, a huge bottleneck still needs to be adressed for the gradient computation. Indeed, we chose to be the most generic possible and use automatic differentiation in order to quickly be able to use a new model and/or a new control problem, but derivating through the junctions operators is graph-dependant which makes this task more complex.

While this approach works well on our test-cases, it still relies too much on the hardward capacity of the user with several GiB allocated through the process. The main reason is the way Julia and automatic differentiation work, and more specifically the conflict between generic user code and specialised (and therefore fast) machine code. This is the topic of the next paragraph.

5.1 Compiler v/s Interpreter

It is important to understand the concepts of compilation and interpretation. A compiler translates human-readable code, the source code, into machine-readable code (assembler/binary) that can be executed later without having to re-read the source code. The interpreter does this translation line-by-line during the execution of the program, which allows greater flexibility since the types can depend on runtime values, but is much slower because less optimisation is possible. There are actually intermediate methods of compilation/interpretation, and Julia uses one of them, called Just-In-Time (JIT) compilation. It is a kind of compromise between the first two methods: the source is dynamically compiled into optimised machine code while the program is running.

Specifically, Julia uses LLVM (Low Level Virtual Machine), a compilation platform designed for compile-time and run-time optimization. It first translates the source code into an intermediate representation (LLVM-IR) that is language-independent and statically typed. After a series of analyses and optimizations, LLVM generates native machine code that is hardware-optimized and efficient because it runs without further interpretation.

Because each function call is specialized with respect to its signature, it is possible to write generic callers that are then statically typed at runtime. This is a key feature of the language, called *Multiple Dispatch*. Source code written in a way that allows LLVM to generate this kind of optimized code is said to be *Type Stable* and ensures high performance. Let us illustrate this with some examples.

```
1  function unstable(x)
2      if x<rand()
3          return "negative x"
4      end
```

```
    return x
end
```

This function illustrates a typical case of type instability. In fact, the return type of the function is a number or a string depending on the sign of `x-rand()`, which is only known at runtime. This prevents the compiler from writing specialized code, resulting in a more complicated intermediate representation (LLVM-IR). We see in the llvm code that some instructions are needed to convert the output to the correct type during execution.

```
julia> @code_llvm unstable(-1)
;  @ REPL[11]:1 within `unstable`
define { {}*, i8 } @julia_unstable_544([8 x i8]* noalias nocapture align 8
    dereferenceable(8) %0, i64 signext %1) #0 {
top:
  %thread_ptr = call i8* asm "movq %fs:0, $0", "=r"() #4
  %ppgcstack_i8 = getelementptr i8, i8* %thread_ptr, i64 -8
  %ppgcstack = bitcast i8* %ppgcstack_i8 to {}****
  %pgcstack = load {}***, {}**** %ppgcstack, align 8
;  @ REPL[11]:2 within `unstable`
; ┌ @ /cache/build/default-amdci4-3/julialang/julia-release-1-dot-8/usr/share/
    julia/stdlib/v1.8/Random/src/Random.jl:257 within `rand` @ /cache/build/
    default-amdci4-3/julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1
    .8/Random/src/Random.jl:257 @ /cache/build/default-amdci4-3/julialang/julia
    -release-1-dot-8/usr/share/julia/stdlib/v1.8/Random/src/Xoshiro.jl:210 @ /
    cache/build/default-amdci4-3/julialang/julia-release-1-dot-8/usr/share/
    julia/stdlib/v1.8/Random/src/Random.jl:257 @ /cache/build/default-amdci4-3/
    julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1.8/Random/src/
    Xoshiro.jl:127
; │┌ @ task.jl:181 within `getproperty`
    %2 = getelementptr inbounds {}**, {}*** %pgcstack, i64 -5
    %3 = bitcast {}*** %2 to i64*
    %4 = load i64, i64* %3, align 8
    %5 = getelementptr inbounds {}**, {}*** %pgcstack, i64 -4
    %6 = bitcast {}*** %5 to i64*
    %7 = load i64, i64* %6, align 8
    %8 = getelementptr inbounds {}**, {}*** %pgcstack, i64 -3
    %9 = bitcast {}*** %8 to i64*
    %10 = load i64, i64* %9, align 8
    %11 = getelementptr inbounds {}**, {}*** %pgcstack, i64 -2
    %12 = bitcast {}*** %11 to i64*
    %13 = load i64, i64* %12, align 8
; │└
; │ @ /cache/build/default-amdci4-3/julialang/julia-release-1-dot-8/usr/share/
    julia/stdlib/v1.8/Random/src/Random.jl:257 within `rand` @ /cache/build/
    default-amdci4-3/julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1
    .8/Random/src/Random.jl:257 @ /cache/build/default-amdci4-3/julialang/julia
    -release-1-dot-8/usr/share/julia/stdlib/v1.8/Random/src/Xoshiro.jl:210 @ /
    cache/build/default-amdci4-3/julialang/julia-release-1-dot-8/usr/share/
    julia/stdlib/v1.8/Random/src/Random.jl:257 @ /cache/build/default-amdci4-3/
    julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1.8/Random/src/
    Xoshiro.jl:128
; │┌ @ int.jl:87 within `+`
    %14 = add i64 %13, %4
; │└
; │ @ /cache/build/default-amdci4-3/julialang/julia-release-1-dot-8/usr/share/
    julia/stdlib/v1.8/Random/src/Random.jl:257 within `rand` @ /cache/build/
```

```
       default-amdci4-3/julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1
       .8/Random/src/Random.jl:257 @ /cache/build/default-amdci4-3/julialang/julia
       -release-1-dot-8/usr/share/julia/stdlib/v1.8/Random/src/Xoshiro.jl:210 @ /
       cache/build/default-amdci4-3/julialang/julia-release-1-dot-8/usr/share/
       julia/stdlib/v1.8/Random/src/Random.jl:257 @ /cache/build/default-amdci4-3/
       julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1.8/Random/src/
       Xoshiro.jl:129
; ┌ @ int.jl:365 within `|`
    %15 = call i64 @llvm.fshl.i64(i64 %14, i64 %14, i64 23)
; └
; ┌ @ int.jl:87 within `+`
    %16 = add i64 %15, %4
; └
; @ /cache/build/default-amdci4-3/julialang/julia-release-1-dot-8/usr/share/
       julia/stdlib/v1.8/Random/src/Random.jl:257 within `rand` @ /cache/build/
       default-amdci4-3/julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1
       .8/Random/src/Random.jl:257 @ /cache/build/default-amdci4-3/julialang/julia
       -release-1-dot-8/usr/share/julia/stdlib/v1.8/Random/src/Xoshiro.jl:210 @ /
       cache/build/default-amdci4-3/julialang/julia-release-1-dot-8/usr/share/
       julia/stdlib/v1.8/Random/src/Random.jl:257 @ /cache/build/default-amdci4-3/
       julialang/julia-release-1-dot-8/usr/share/julia/stdlib/v1.8/Random/src/
       Xoshiro.jl:130
; ┌ @ int.jl:503 within `<<` @ int.jl:496
    %17 = shl i64 %7, 17
; └

#=

                                [+ 100 lines...]

=#

; ┌ @ float.jl:310 within `unsafe_trunc`
    %31 = fptosi double %27 to i64
    %32 = freeze i64 %31
; └
; @ float.jl:454 within `<` @ int.jl:83
   %33 = icmp sgt i64 %32, %1
; @ float.jl:454 within `<`
; ┌ @ bool.jl:39 within `|`
    %34 = or i1 %30, %33
; └
; ┌ @ bool.jl:38 within `&`
    %35 = and i1 %34, %29
; └
; ┌ @ bool.jl:39 within `|`
    %36 = or i1 %28, %35
; └└
  br i1 %36, label %common.ret, label %L56

common.ret:                                       ; preds = %L56, %top
  %common.ret.op = phi { {}*, i8 } [ { {}* null, i8 1 }, %L56 ], [ { {}*
      inttoptr (i64 139965076172720 to {}*), i8 -128 }, %top ]
;  @ REPL[11] within `unstable`
  ret { {}*, i8 } %common.ret.op
```

```
73
74 L56:                                                ; preds = %top
75 ;  @ REPL[11]:5 within `unstable`
76   %.0..sroa_cast = bitcast [8 x i8]* %0 to i64*
77   store i64 %1, i64* %.0..sroa_cast, align 8
78   br label %common.ret
79 }
```

The following is a stable variant of this function, where zero(x) returns 0 of the same type as x.

```
function stable(x)
    if x<0
        return "negative x"
    end
    return x
end
```

A look at the llvm code, for example for `stable(-1)`, shows that there are fewer instructions and most importantly that the type is now a compilation constant.

```
julia> @code_llvm stable(-1)
;  @ REPL[17]:1 within `stable`
define { {}*, i8 } @julia_stable_546([8 x i8]* noalias nocapture align 8
    dereferenceable(8) %0, i64 signext %1) #0 {
top:
;  @ REPL[17]:2 within `stable`
; ┌ @ int.jl:83 within `<`
   %2 = icmp sgt i64 %1, -1
; └
  br i1 %2, label %L4, label %common.ret

common.ret:                                           ; preds = %L4, %top
  %common.ret.op = phi { {}*, i8 } [ { {}* null, i8 1 }, %L4 ], [ { {}* inttoptr
       (i64 139965083192880 to {}*), i8 -128 }, %top ]
;  @ REPL[17] within `stable`
  ret { {}*, i8 } %common.ret.op

L4:                                                   ; preds = %top
;  @ REPL[17]:5 within `stable`
  %.0..sroa_cast = bitcast [8 x i8]* %0 to i64*
  store i64 %1, i64* %.0..sroa_cast, align 8
  br label %common.ret
}
```

Actually, this code is still inconvenient because the function's return type is `Union{Int64, String}`, which is a data type that can contain two primitive types. By setting a single primitive return type for the function, the following code solves this problem.

```
function stable(x)
    if x<0
```

```
3         return zero(x)
4     end
5     return x
6 end
```

The corresponding LLVM translation is now really short and also fairly human readable:

```
1 julia> @code_llvm stable(-1)
2 ;  @ REPL[10]:1 within `stable`
3 define i64 @julia_stable_532(i64 signext %0) #0 {
4 top:
5   %1 = icmp sgt i64 %0, 0
6   %. = select i1 %1, i64 %0, i64 0
7 ;  @ REPL[10] within `stable`
8   ret i64 %.
9 }
```

Essentially, it means that:

1. The register `%1` stores the value of the comparison between the register `%0` of type `i64` (integer on 64 bits) and `0`.

2. Then, the register `%.` is filled with the value stored in `%0` or `0`, both of type `i64`, depending on the value of `%1`.

3. The last line shows that the returned value is of type `i64`.

The key point to remember herewith is that it is advantageous to have type-stable code in order to reduce the size of the LLVM code and hence its execution time.

5.2 Accurate numerical differentiation within floating-point numbers arithmetic

Before actually talking about automatic differentiation, it is primordial to explain the drawbacks of the floating-points arithmetic when it comes to compute accurate local growth rate of a function.

The first approach that comes to mind is, for a function $x \mapsto f(x)$, to consider its derivative in some x_0 to be:

$$\frac{f(x_0+\varepsilon)-f(x_0)}{\varepsilon},$$

for a small $\varepsilon > 0$. A Taylor expansion tells us that this expression yields $f'(x_0) + O(\varepsilon)$ wherever f is C^1, so we want ε to be as small as possible, especially since errors add up when composing the derivatives. But if this reasonning is relevant in a mathematical framework, it simply causes the derivative to vanish at some point, because there is no guarantee that $x_0 + \varepsilon$ and x_0 will be encoded as two differents numbers.

Floating-points arithmetic. The fact that there are infinitely many numbers and only a finite number of transistors implies that not all real numbers can be represented, and that only a certain number of decimals are

significant. For instance, a number like 0.1 is expressed internally in the following way:

$$0.1 \rightsquigarrow \underbrace{+}_{\text{sign}} \underbrace{1.00000000}_{\text{mantissa}} \times \underbrace{10^{-1}}_{\text{exponent}},$$

and inaccuracy occurs when an operation is performed between two numbers of very different orders of magnitude, as illustrated Table 3.2, where we write the successive operations used to compute $10^5 + 0.066567896$.

Operation	Remark
$1.00000000 \times 10^5 + 0.066567896$	
$= 1.00000000 \times 10^5 + 0.00000066 \times 10^5$	Convertion to the same exponent
$= 1.00000066 \times 10^5$	Loss of information

Table 3.2: Loss of information using floating-points arithmetic.

The consequence on differentiation is that, for ε equal to the machine precision,

$$x_0 + \varepsilon - x_0 = \begin{cases} 0, & \textit{Float}, \\ \varepsilon, & \textit{Maths}. \end{cases}$$

The solution of the forward automatic differentiation implemented in the Julia package `ForwardDiff` is to separate x_0 and ε along two dimensions, using the so-called dual Numbers.

Dual Numbers. The dual numbers are a two-dimensional commutative unitary associative algebra over the real numbers, arising from the reals by addition of a new element ε with the property $\varepsilon^2 = 0$ (ε is a nilpotent element). They were introduced by William Clifford in 1873. Such a number writes uniquely as $z = a + \varepsilon b =: (a, b)$, for some $a, b \in \mathbb{R}$. Their use in automatic differentiation is straightforward, because for these numbers Taylor expansions yields:

$$f(x_0 + \varepsilon h) = f(x_0) + f'(x_0)h\varepsilon + O(\varepsilon^2) = f(x_0) + f'(x_0)h\varepsilon = \big(f(x_0), f'(x_0)h\big),$$

meaning that we compute both the value and the derivative of the function in one evaluation, in an exact way.

This framework readily extends in higher dimensions into multi-duals:

$$\begin{aligned} &\forall i, j \quad \varepsilon_i \varepsilon_j = 0, \\ &f(x_0 + \varepsilon) = f(x_0) + \nabla f(a) \cdot \varepsilon = \big(f(x_0), \nabla f(x_0)\big) \end{aligned}$$

We can again compute ∇f in a single (vector) evaluation instead of computing $\partial_i f$ separately.

5.3 Towards type-stable automatic differentiation in Julia, conclusion and perspectives

The use of ForwardDiff has many advantages in terms of implementation (one line only to compute a Jacobian of composed vector functions) but introduces a high execution cost, as we are in a case where type-stability and thus pre-allocations are more difficult to obtain simultaneously. We gain in simplicity of use but the optimization of the code requires a real understanding of some internal mechanisms of Julia.

Although the current version of the code is satisfactory from a research point of view (a few minutes to run test cases of one or two dozens of roads), it is still very insufficient if we envisage a scale-up to an industrial environment, with test cases of an entire city with several hundred or even thousands of roads to simulate.

After profiling the code, it is clear that most of the computation time is spent computing the gradient of the linear programming problem, which is not surprising since the graph structure is used extensively.

Beyond the necessary considerations of parallelization, it seems that large gains in execution time and memory are achievable in a first step by reaching a suitable mix between static pre-allocations and dynamic allocations. In the first case the initial launch cost is increased for the benefit of a faster execution, contrary to the second case where the initial cost is lower at the cost of an overload of the execution time. In this case, it is not guaranteed that the hardware supports too many simultaneous allocations, but a too slow execution of the optimization algorithm is also undesirable, more specifically in a crisis management context.

The natural perspective that emerges from this is to achieve real-time optimal control over large networks, such as thousands of edges for a city like Paris. Amongst all tools that can be used to achieve this, graph reduction and reduced order models seems to be of great interest and the later is investigated in chapter 5 which is devoted to the hyper-reduction of the finite volumes scheme on a network.

Part II

Numerical Schemes for Strongly Nonlinear Fluid Dynamics

Chapter 4

Tools for ROMs and ML

Eigenmode decomposition simplifies data by preserving the essential structure and removing redundancies. This allows complex equations to be transformed into a compact, reliable system by means of a well-chosen projection. However, in the presence of strong non-linearity, conventional methods such as EIM and DEIM run into difficulties. We propose to address this issue using neural networks. After a brief introduction to the latter, the chapter ends with a presentation of stochastic gradient descent. This is an efficient approach to high-dimensional optimization that lies behind methods such as Adam and RMS prop, which are commonly found in standard machine learning libraries.

1 Reduced Order Modeling

The numerical solution of optimal control problems involves a large number of calls to the solver of the physical system under consideration, as with the direct adjoint looping (DAL) method [49] presented in chapter 2. In order to reduce the computation time associated with these methods, we are interested in the possibility of capturing most of the dynamics with as few degrees of freedom as possible. A common approach to achieving this goal is to use the Reduced Order Model (ROM) framework. This framework is designed to create a compact yet precise model capable of accurately representing a range of parameter variations. The ultimate goal is to enable real-time simulation of parameterized models. This approach is rooted in the recognition that natural phenomena typically exhibit an inherent structure that allows them to be effectively characterized by the superimposition of their most significant patterns. Our approach is based on a priori data from full-order models (FOMs), such as finite elements or finite volumes, which we use to construct an appropriate low-rank subspace onto which we project the dynamics. However, this method quickly runs into difficulties due to significant nonlinearities. To solve this problem, we will develop an adapted closure mechanism to efficiently handle projection errors. Our approach is to exploit the remarkable approximation capabilities of a neural network through a data-driven strategy.

See [1, 89] for a comprehensive review of this topic. We're now going to introduce some of the tools that we will need in the following.

1.1 Best low-rank approximation

Any matrix $A \in \mathcal{M}_{n,m}(\mathbb{R})$ admits a proper orthogonal decomposition given via singular values decomposition (SVD) due to the fact that we can always apply the spectral theorem to the symetric positive definite matrices

A^TA and AA^T. This yields the following theorem:

Theorem 9. *(SVD) Let $A \in \mathcal{M}_{n,m}(\mathbb{R})$ of rank $p \leq \min\{n,m\}$. Then there exists positive real numbers $\sigma_1 \geq \cdots \geq \sigma_p > 0$, and orthogonal matrices $U \in \mathcal{O}_n(\mathbb{R}), V \in \mathcal{O}_m(\mathbb{R})$, such that:*

$$A = U\Sigma V^T, \quad \Sigma = \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix}, \quad D = \mathrm{diag}(\sigma_1, \cdots, \sigma_p). \tag{4.1}$$

Moreover, this theorem is immediately related to our compression concern, since it allows us to approximate any matrix by one of lower rank:

Theorem 10. *Eckart-Young (1936) Let $A \in \mathcal{M}_{n,m}(\mathbb{R})$ of rank p. Then, for any $r \leq p$:*

$$\min_{rg(X)=r} \|A - X\|_F = \|A - A_r\|_F, \tag{4.2}$$

where $A_r = U_r\Sigma_r V_r^T$, with U_r, Σ_r and V_r^T being the r-th firsts columns of U, Σ and V^T respectively.

1.2 POD based Reduced Order Modeling (POD-ROM)

Let us consider a system of ODEs in high dimension $N_x >> 1$, obtained for instance from the discretization of a PDE and indexed by a family of parameters $\mu \in \mathbb{R}$:

$$\frac{dx}{dt}(t;\mu) = F(x(t;\mu);\mu), \quad x(t;\mu) \in \mathbb{R}^{N_x}. \tag{4.3}$$

In order to reduce the dimension, we make the fundamental hypothesis that the high dimensionnal trajectory $x(t;\mu)$ can be represented by a trajectory $\hat{x}(t;\mu)$ that lies on a low-dimensionnal hyperplane, i.e. there exists a vectorial subspace $\mathcal{V} := \mathrm{span}\{\phi_j\}_{j=1}^r$ with $r << N_x$, and $x^{ref} \in \mathbb{R}^{N_x}$ such that:

$$x(t;\mu) \approx x^{ref} + \Phi\hat{x}(t;\mu), \quad \Phi = \begin{bmatrix} | & & | \\ \phi_1 & \cdots & \phi_r \\ | & & | \end{bmatrix} \in \mathbb{R}^{N_x \times r}. \tag{4.4}$$

We will also call Φ a *decoder* since it maps a low-rank solution $\hat{x} \in \mathbb{R}^r$ onto the full dimensional space $\mathbb{R}^{N_x}$. From now on, we will suppose without loss of generality that $x^{ref} \equiv 0$, for the sake of readability.

Remark 6. *Note that the fundamental hypothesis above assumes some linearity in the behavior of the solution, thus will not hold in non-linear cases. This is the topic of subsection 1.3.*

We apply the SVD, which is named the *proper orthogonal decomposition* (POD) in this context, on snapshots of the full model (4.3) to find an orthogonal basis on which to express the solution, such that most of the useful information is stored in the first low-frequency coefficients. The snapshots are data considered to be of *high-fidelity* since they are computed using the full order model on a fine mesh.

The purpose of the POD is then to determinate a low-rank approximation basis, a decoder, Φ from the datum $x(t;\mu)$. To this end, we use the approach of [47] that is to form the snapshots matrix:

$$\mathcal{X} := \left(x(t_1,\mu_1), \cdots, x(t_{N_t},\mu_1), \cdots, x(t_1,\mu_{N_\mu}), \cdots, x(t_{N_t},\mu_{N_\mu})\right),$$

which is of size $N_x \times N_\mu N_t$. We then look for any $K \leq r$ for a decoder Φ_K that solves:

$$\min_{\Phi_K^T \Phi_K = id} \left\| \mathcal{X} - \Phi_K \Phi_K^T \mathcal{X} \right\|_F.$$

By Theorem 10, the solution is given by the K firsts columns of the eigen vectors of the SVD decomposition:

$$\Phi_K := (U_1 \cdots U_K), \quad \mathcal{X} = U \Sigma V^T. \tag{4.5}$$

Remark 7. *The numbers σ_i, called the* singular values *of $\mathcal{X}$, are equal to the eigenvalues of both $\mathcal{X}^T\mathcal{X}$ and $\mathcal{X}\mathcal{X}^T$, and are ordered in decreasing order: $\sigma_1 \geq \cdots \geq \sigma_p \geq 0$. To make an analogy with Fourier series, note that the eigenvectors associated with the first eigenvalues are called* low-frequency modes. *Ignoring the information provided by the smallest eigenvalues is then similar to using a low-pass filter.*

With this construction we now are able to project (4.3) on the hyperplane generated by the snapshots. Writing $\hat{x} := \Phi^T x$, we now consider:

$$\frac{d\hat{x}}{dt}(t;\mu) = \Phi^T F(x(t;\mu);\mu), \tag{4.6}$$

that is approximated, using our fundamental hypothesis, by:

$$\frac{d\hat{x}}{dt}(t;\mu) = \Phi^T F(\Phi\hat{x}(t;\mu);\mu). \tag{4.7}$$

In the case where F is linear, we are then led to solve:

$$\frac{d\hat{x}}{dt}(t;\mu) = \left(\Phi^T F \Phi\right)(\hat{x}(t;\mu);\mu). \tag{4.8}$$

We perform a single off-line calculation of the full-dimensional matrix, denoted as $L := \Phi^T F \Phi$. Subsequent numerical integration exclusively operates within the reduced dimension. During the online phase, for example using explicit Euler integration on a regular grid $(t_n)_{1\leq n\leq N_t}$ and writing $\hat{x}_\mu^n := \hat{x}(t_n;\mu)$, we compute:

$$\hat{x}_\mu^{n+1} = \hat{x}_\mu^n + \Delta t L \hat{x}_\mu^n = (I + \Delta t L)\hat{x}_\mu^n, \quad 1 \leq n \leq N_t - 1. \tag{4.9}$$

A nonlinear function F leads to the obligation to compute the high dimensional expression $F(\Phi\hat{x}_\mu^n)$ at each time step. This is obviously not efficient in a reduction framework. An approach to deal with this case is presented in the next section.

1.3 Nonlinear case: DEIM

The nonlinarity of the flux makes the reduced basis from snapshots inneficient to compress the trajectory of the system because the evaluation of the nonlinearity costs as mush as the resolution of the high fidelity model. The construction of an actual low rank evaluation of the non linear flux that we present here is inspired by what is often known as *sparse sampling* [35] and relies on the statement that meaningful data are sparse in the set of all possibles data, i.e. natural data are compressible. From that, one can infer that a suitable configuration of sparse interpolations points (sometimes refered to as *sensors*) allows an accurate reconstruction of the signal without needing the entier sampling of the high fidelity data of reference. This principle has a lot of applications

[67, 68, 96] and we use it here to sparsely sample the nonlinearity in the ODE. This is achieved thanks to the low-rank structure of the flux that we capture via a second SVD. We first reduce the flux separately, which is called *hyper-reduction* [85]. We build the flux snapshots:

$$\mathcal{N} := \left(F(x(t_1,\mu_1);\mu_1),\cdots,F(x(t_{N_t},\mu_1);\mu_1),\cdots,F(x(t_1,\mu_{N_\mu});\mu_{N_\mu}),\cdots,F(x(t_{N_t},\mu_{N_\mu});\mu_{N_\mu})\right),$$

that we also decompose using SVD:

$$\mathcal{N} = \Xi \Lambda W^T,$$

where $\Xi = (\xi_i)$ and $W = (w_j)$ are orthogonal, and $\Lambda = (\lambda_k)$ the matrix of corresponding singular values. The information given by this new modal basis allows us to construct a sparse sampling matrix $P \in \mathbb{R}^{p \times N_x}$ thanks to a greedy algorithm that selects the p sampling locations such as maximizing recursively the projection error between ξ_k and $[\xi_1,\cdots,\xi_{k-1}]$, for $k \leq p$. More details about the construction of P and further considerations can be found in the work of [56]. The sparse sampling matrix P is then used to construct the *oblic projector* of rank $p << N_x$:

$$\Pi_{ob} := \Xi_p \left(P^T \Xi_p\right)^{-1} P^T,$$

such that we now replace the costly resolution of Equation 4.7 by:

$$\frac{d\hat{x}}{dt}(t;\mu) = \Phi^T \Pi_{ob} F(\Phi \hat{x}(t;\mu)). \tag{4.10}$$

The advantage of this formulation is that the flow evaluation now requires only $p << N_x$ evaluation of the nonlinearity, at the coordinates selected by the projector.

2 Deep learning tools

The last section of this chapter introduces classical considerations in deep learning, which is a tool that is more and more mixed with usual numerical analysis [69] and is now extensively used in various fields such as image processing, language analysis or fast PDE resolution. We first introduce the overall context of neural networks with a focus on multilinear perceptrons (MLP). Then, we give some insights about the backbone of the numerical feasibility of those methods, the so-called stochastic gradient descent (SGD).

2.1 Neural Networks

A neural network is a parametric non-linear function, that is obtained as the composition of linear weighted functions with nonlinear (activation) functions denoted σ. The classic way of representing this is by stacking layers together:

Input layer	z^0,	
Hidden layers	$z^k = \sigma\left(W^k z^{k-1} + b^k\right)$,	$1 \leq k \leq l-1$
Output layer	$z^l = W^l z^{l-1} + b^l$,	

where we call $\theta := \{W^k, b^k\}_{1\leq k\leq l}$ the trainable parameters, and activation functions $\sigma(\cdot)$ are usually choosen amongst tanh, ReLU, softmax, ...

Neural networks provide a very effective tool for approximating functions, and its use can be justified theoretically by numerous universal approximation theorems of which we give here one of the first formulations by Cybenko:

Theorem 11. *Universal approximation, [28, Cybenko, 1989]. The set of feed-forward neural networks is dense in the set of functions that are continuous on a given compact, for the uniform topology:*
Writing K a compact subset of $\mathbb{R}^n$ and defining

$$\mathcal{NN} = \{x \mapsto C \cdot (\sigma \circ (W \cdot x + b));\ A \in \mathcal{M}_{k,n}(\mathbb{R}), b \in \mathbb{R}^k, C \in \mathcal{M}_{m,k}(\mathbb{R})\},$$

then

$$\overline{\mathcal{NN}} = \mathcal{C}(K; \mathbb{R}^m), \quad \textit{for } \|\cdot\|_{\infty,K}.$$

There are refinements of this theorem which give the same kind of results for less regular functions, such as Bochner-Lebesgue integrable functions, and for bounded networks with a control of the dimension, see [39] for a review. Although other architectures have proved highly effective for certain classes of problems (CNN in image processing, RNN for time series, transformers for language processing, ...), we will restrict ourselves to the class introduced above.

To find the optimal trainable parameters $\theta = \{W^k, b^k\}$, we perform a gradient-based optimization of a loss function $\mathcal{L}$ that mainly quantifies the data attachment of the parameterized model, as well as regularization criteria. A prototypical example of such methods is explained in the next subsection.

2.2 Stochastic Gradient Decent

A central element in the implementation of neural networks is the ability to automatically and accurately compute high-dimensional gradients. Automatic differentiation being the subject of subsection 5.2, we introduce here the essential tool for the practical feasibility of high-dimensional computations: the stochastic gradient descent [84]. We begin by discussing its theoretical construction before presenting the algorithms that enable it to be used in practice.

Continuous problem: Let $(\Omega, \mathcal{A}, \mathbb{P})$ be a probability space, $(\mathcal{U}, <\cdot,\cdot>)$ be a Hilbert space whose norm is denoted by $\|\cdot\|$, and $U^{ad} \subset \mathcal{U}$ be a closed convex subset. Let $W : \Omega \to \mathcal{W}$ be a random variable, and $j : \mathcal{U} \times \mathcal{W} \to \mathbb{R}$ a function that is differentiable in its first variable and integrable in the second:

$$\begin{aligned} &\forall w \in \mathcal{W}, \quad j(\cdot, w) \text{ is differentiable,} \\ &\forall u \in U^{ad}, \quad j(u, \cdot) \in L^1(\Omega, \mathcal{A}, \mathbb{P}). \end{aligned}$$

The generic problem of interest is the following minimization problem:

$$(\mathcal{P}) : \inf_{u\in U^{ad}} J(u), \qquad \text{where } J : \begin{array}{ccc} U^{ad} & \to & \mathbb{R} \\ u & \mapsto & \mathbb{E}_W(j(u, W)) \end{array}. \tag{4.11}$$

Writting $\mu := \mathbb{P} \circ W^{-1}$ the pushforward measure of $\mathbb{P}$ by W, we have:

$$\mathbb{E}_W(j(u,W)) = \int_\Omega j(u,W(\omega))\, d\mathbb{P}(\omega) = \int_{\mathcal{W}} j(u,w)\, d\mu(w).$$

By differentiability of j and derivation under the integral sign, we also have :

$$\forall u \in U^{ad}, \quad \nabla J(u) = \mathbb{E}_W(\nabla_u j(u,W)). \tag{4.12}$$

Approximated problem: The numerical approximation of the previous problem leads us to consider the following approximate Monte-Carlo problem:

$$(\mathcal{P}_{app}) : \inf_{u\in U^{ad}} J_{app}, \qquad \text{where } J_{app} : \begin{array}{ccc} U^{ad} & \to & \mathbb{R} \\ u & \mapsto & \frac{1}{N}\sum_{i=1}^N j(u,w^i) \end{array}, \tag{4.13}$$

where $\left(w^1, \cdots, w^m\right)$ is the realization of an i.i.d. m-sample of W, i.e. an i.i.d. sequence $\left(W^1, \cdots, W^m\right)$ where $\forall i$, W^i has the same distribution as W.

We therefore obtain the following estimate:

$$\nabla J(u) \simeq \nabla J_{app}(u) = \frac{1}{N}\sum_{i=1}^N \nabla_u j(u,w^i). \tag{4.14}$$

This formulation is not numerically advantageous, as it involves calculating a large number of gradients at each step, N being in machine learning the size of the dataset. It is indeed conventional [25] to consider a projected gradient descent algorithm:

$$u^0 \text{ given}, \qquad u^{k+1} \leftarrow \Pi_{U^{ad}}\left(u^k - \delta_k \nabla J_{app}(u)\right), \tag{4.15}$$

involving the calculation of m gradients, possibly in high dimension, at each iteration k.

To address this problem, we will avoid calculating the expectation along W by randomly picking a sample following the law of W at each iteration, forming a sequence of estimators converging to the solution of the problem:

STOCHASTIC GRADIENT DESCENT ALGORITHM (SGD)

1. **Initialization.**
   $k = 0$: chose $u^0 \in \mathcal{U}$ and $\rho_0 \in \mathbb{R}_+^*$.
2. **Iteration k.** Pick a sample w^{k+1} from the random variable W and update:
$$u^{k+1} = \Pi_{U^{ad}}\left(u^k - \delta_k \nabla_u j(u^k, w^{k+1})\right).$$

We notice that the stochastic gradient $\nabla_u j(u^k, w)$ is an unbiased estimator of the deterministic gradient $\nabla J(u)$ according to (4.12).

To improve convergence, we use the stochastic gradient "minibatch", which consists of randomly drawing a "batch", i.e. an n-arrangement in $[\![1, N]\!]^n$, calculating the gradient using the n corresponding realizations of the

w_i, and repeating until all the training data have been run through. Then start again.

SGD MINI-BATCH

1. **Initialization.**
   $k = 0$: chose $u^0 \in \mathcal{U}$ and $\delta_0 \in \mathbb{R}_+^*$.
2. **Iteration k.**
   - We draw w^{k+1} from W, which we partition into N/n batches $(b_t)_t$ of size n.
   - batches iterations:

$$u^{t+1} \leftarrow u^t - \delta_t \frac{1}{|b_t|} \sum_{i \in b_t} \nabla_u j(u^t, i).$$

Chapter 5

Comparative approaches to hyper-reduction using deep learning

1 General framework

In order to use large-scale modeling tools, for example in a control context where the resolution of a large nonlinear system is repeatedly required, it is relevant to consider order reduction. Such methods aim to simulate complex models in real time by appropriately compressing their degrees of freedom. Morally, the approach is close to what is classically done with Fourier series, where the unknown is decomposed on a modal basis so as to preserve the essential dynamics once the high frequencies have been cut off:

$$\rho(t) \approx \sum_{k=0}^{N_m} a_k(t)\phi_k.$$

In our work, the expansion basis will be created from data generated by the model, called *snapshots*.

For our purpose, we want to solve congestion problems in a large city, involving solving the LWR model several times on a large road network in order to use the optimal control tools introduced in chapter 2. The system can be parameterized with respect to quantities such as maximum velocities $(v_i^{\max})_i$, maximum densities

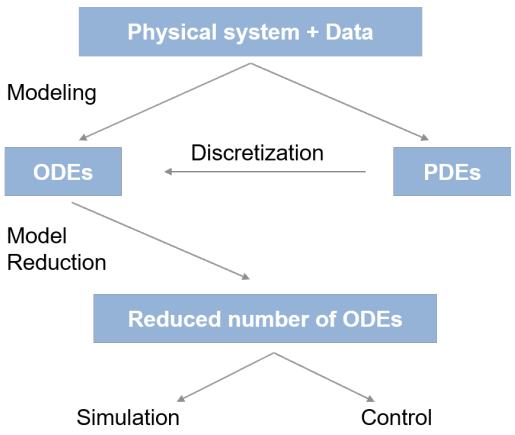


Figure 5.1: By Slevin48an209 - CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=95887101

$(\rho_i^{\max})_i$, or initial data $(\rho_{i,0})_i$. For the sake of generality, we will refer to such parameters as $\boldsymbol{\mu}$ in all that follows. For more details on this model, please refer to chapter 2.

$$\begin{cases} \partial_t \rho_i(t,x;\boldsymbol{\mu}) + \partial_x f_i(\rho_i(t,x;\boldsymbol{\mu});\boldsymbol{\mu}) = 0, & (t,x) \in (0,T) \times [a_i,b_i], \\ \rho_i(0,x;\boldsymbol{\mu}) = \rho_{i,0}(x;\boldsymbol{\mu}), & x \in [a_i,b_i], \end{cases}, \quad 1 \leq i \leq N_r.$$

After a spatial discretization using N_c cells per road, thus writing $N_x := N_r N_c$ the total degrees of freedom of the parameterized numerical unknown $\boldsymbol{\rho}(\cdot\,;\boldsymbol{\mu}) \in \mathbb{R}^{N_x}$ on the network, we obtain the following system of ODEs:

$$\begin{cases} \frac{d\boldsymbol{\rho}}{dt}(t;\boldsymbol{\mu}) = \mathbf{F}\left(\boldsymbol{\rho}(t;\boldsymbol{\mu});\boldsymbol{\mu}\right), & t \in (0,T), \\ \boldsymbol{\rho}(t=0;\boldsymbol{\mu}) = \boldsymbol{\rho}^0_{\boldsymbol{\mu}} \in \mathbb{R}^{N_x}. \end{cases} \tag{5.1}$$

Remark 8. *We need a fairly large number of cells per route to obtain a high-fidelity model, which will serve as a baseline in our investigations.*

The goal now is to build a surrogate model with an approached solution $\hat{\boldsymbol{\rho}} \in \mathbb{R}^{N_m}$ associated to a low rank flux $\hat{\mathbf{F}}(\hat{\boldsymbol{\rho}})$, with $N_m << N_x$, on the form:

$$\frac{\mathrm{d}\hat{\boldsymbol{\rho}}}{\mathrm{d}t} = \hat{\mathbf{F}}(\hat{\boldsymbol{\rho}}), \quad \mathrm{dist}(\hat{\boldsymbol{\rho}},\boldsymbol{\rho}) \text{ is small.} \tag{5.2}$$

We aim to identify an appropriate low-rank subspace that can provide an accurate solution representation. In line with the POD-ROM framework, our goal is to develop an algorithm that divides the process into offline and online phases. During the offline phase, we create this subspace using prior data, while the online phase exclusively involves low-dimensional computations.

To achieve this, we employ the snapshot approach, wherein we calculate the full-order solution for a set comprising N_μ parameters $\mathcal{P}_\mu := \{\boldsymbol{\mu}_k,\ 1 \leq k \leq N_\mu\}$ and N_t time steps $\mathcal{P}_t := \{t_n,\ 1 \leq n \leq N_t\}$ to generate the snapshot matrix $S \in \mathcal{M}_{N_x,N_\mu N_t}(\mathbb{R})$. Writing $N_s := N_\mu \times N_t$ the number of snapshots and $\rho^{j,n}_{\boldsymbol{\mu}_k} = \rho_j(t_n;\boldsymbol{\mu}_k)$, we have:

$$S := \begin{pmatrix} \rho^{1,1}_{\boldsymbol{\mu}_1} & \cdots & \rho^{1,N_t}_{\boldsymbol{\mu}_1} & \cdots & \cdots & \cdots & \rho^{1,1}_{\boldsymbol{\mu}_{N_\mu}} & \cdots & \rho^{1,N_t}_{\boldsymbol{\mu}_{N_\mu}} \\ \vdots & & & & & & & & \vdots \\ \vdots & & & & & & & & \vdots \\ \vdots & & & & & & & & \vdots \\ \rho^{N_x,1}_{\boldsymbol{\mu}_1} & \cdots & \rho^{N_x,N_t}_{\boldsymbol{\mu}_1} & \cdots & \cdots & \cdots & \rho^{N_x,1}_{\boldsymbol{\mu}_{N_\mu}} & \cdots & \rho^{N_x,N_t}_{\boldsymbol{\mu}_{N_\mu}} \end{pmatrix}, \tag{5.3}$$

to which we apply a Proper Orthogonal Decomposition (POD):

$$S = U\Sigma V^T,$$

with $U \in \mathcal{O}_{N_x}(\mathbb{R})$, $V \in \mathcal{O}_{N_s}(\mathbb{R})$ and $\Sigma \in \mathcal{M}_{N_x,N_s}(\mathbb{R})$ of rank p such that:

$$\forall 1 \leq i \leq p:\ \Sigma_{i,i} = \sigma_i \geq 0, \quad \forall i > p:\ \Sigma_{i,i} = 0, \quad \forall i \neq j:\ \Sigma_{i,j} = 0.$$

The reduced basis with N_m modes is given by the first N_m columns of the matrix U, i.e.

$$\Phi = \begin{pmatrix} \phi_1 & \cdots & \phi_{N_m} \end{pmatrix} := \begin{pmatrix} U_1 & \cdots & U_{N_m} \end{pmatrix}. \tag{5.4}$$

We use it to apply a Galerkin projection on the high-fidelity system, which yields the following reduced ODE:

$$\begin{cases} \Phi^T \frac{d\boldsymbol{\rho}}{dt}(t;\boldsymbol{\mu}) = \Phi^T \mathbf{F}\left(\boldsymbol{\rho}(t;\boldsymbol{\mu});\boldsymbol{\mu}\right), \quad t \in (0,T), \\ \Phi^T \boldsymbol{\rho}(t=0;\boldsymbol{\mu}) = \Phi^T \boldsymbol{\rho}^0_{\boldsymbol{\mu}} \in \mathbb{R}^{N_m}, \end{cases} \tag{5.5}$$

that we rewrite, using the approximation from the fundamental hypothesis $\Phi\hat{\boldsymbol{\rho}} \approx \boldsymbol{\rho}$:

$$\begin{cases} \frac{d\hat{\boldsymbol{\rho}}}{dt}(t;\boldsymbol{\mu}) = \Phi^T \mathbf{F}\left(\Phi\hat{\boldsymbol{\rho}}(t;\boldsymbol{\mu});\boldsymbol{\mu}\right), \quad t \in (0,T), \\ \hat{\boldsymbol{\rho}}(t=0;\boldsymbol{\mu}) = \hat{\boldsymbol{\rho}}^0_{\boldsymbol{\mu}} \in \mathbb{R}^{N_m}. \end{cases} \tag{5.6}$$

However, the non-linearity of $\mathbf{F}$ raises important questions. Indeed, we can see that evaluating $\Phi^T\mathbf{F}(\Phi\,\cdot)$ has essentially the same cost as $\mathbf{F}$ and moreover needs to be done at each time step, thus obliterating any gain in order reduction. A classical way to deal with this problem is to add an hyper-reduction such as the Dynamic Empirical Interpolation Method (DEIM) presented in subsection 1.3:

$$\frac{d\hat{\boldsymbol{\rho}}}{dt}(t;\boldsymbol{\mu}) = \hat{\mathbf{F}}^{DEIM}(\hat{\boldsymbol{\rho}}(t;\boldsymbol{\mu});\boldsymbol{\mu}). \tag{5.7}$$

However, strong nonlinearity in the flux $\mathbf{F}$ makes it necessary to use an increasing number of modes to get an accurate dynamic. Therefore, we usually add a closure term, which amounts to consider:

$$\frac{d\hat{\boldsymbol{\rho}}}{dt}(t;\boldsymbol{\mu}) = \hat{\mathbf{F}}^{DEIM}(\hat{\boldsymbol{\rho}}(t;\boldsymbol{\mu});\boldsymbol{\mu}) + C_{\boldsymbol{\mu}}(\hat{\boldsymbol{\rho}}(t;\boldsymbol{\mu})), \tag{5.8}$$

and determine an appropriate $C_{\boldsymbol{\mu}}$ that corrects out of the retained modes the error made on the truncated modes. This point of view is widely discussed in [1, 89].

We highlight the limitations of the DEIM approach for genuine nonlinear flow scenarios in Figure 5.2. In this context, we employ the LWR model on two roads as the governing equation, with specific parameters and model details provided in subsection 3.1.

We chose to simultaneously compute a hyper-reduction flux and a closure term to ensure both reliability and stability of the reduction, all within a model-agnostic framework that derives the flux solely from low-dimensional information, as schematically illustrated Figure 5.3. In the following section, we describe our approach to address this challenge.

2 Presentation of our strategy

The approach that we propose in this work is to use a neural network to learn the hyper-reduction and the closure in a same function. This leads us to consider the system:

$$\frac{d\hat{\boldsymbol{\rho}}}{dt}(t;\boldsymbol{\mu}) = \hat{F}_\theta(\hat{\boldsymbol{\rho}}(t;\boldsymbol{\mu}),\boldsymbol{\mu}), \tag{5.9}$$

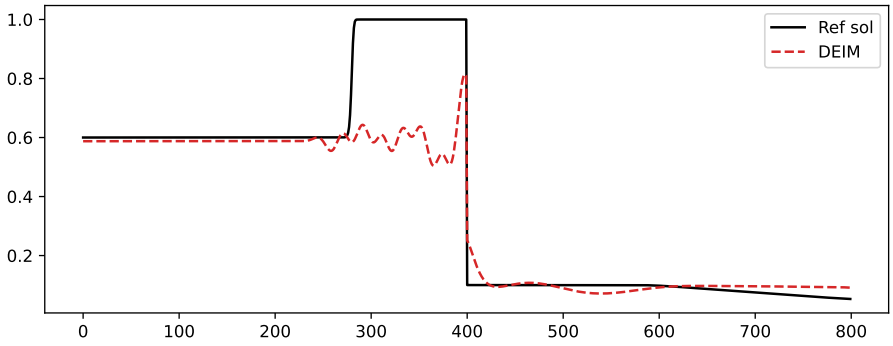


Figure 5.2: Reference solution (Finite Volumes) and DEIM reconstruction.

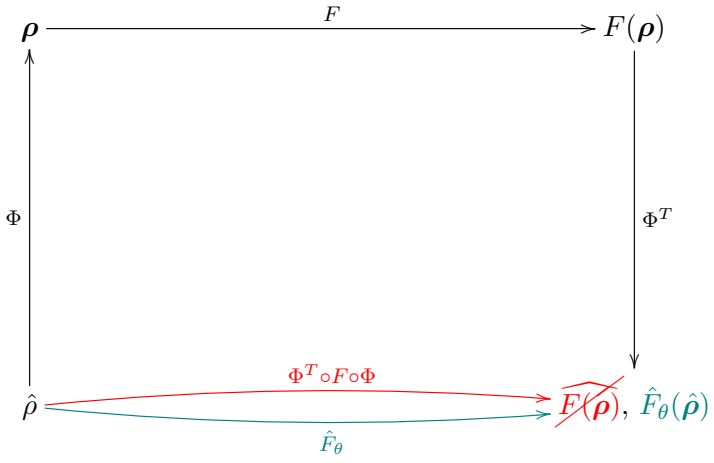


Figure 5.3: Objective of the deep hyper-reduction method.

where $\hat{F}_\theta$ is a neural network with weights $\theta = (\theta_1, \cdots, \theta_p)$ that takes $(\hat{\boldsymbol{\rho}}(t_n;\boldsymbol{\mu}), \boldsymbol{\mu})$ as input and returns a prediction of the compressed flux $\hat{F}_\theta(\hat{\boldsymbol{\rho}}(t_n;\boldsymbol{\mu}), \boldsymbol{\mu}) \approx \Phi^T \mathbf{F}(\boldsymbol{\rho}(t_n;\boldsymbol{\mu});\boldsymbol{\mu})$. We see that this network is precisely designed to recover the benefits of a reduction, since the online stage uses the reduced dimension only, see Figure 5.3. Note that learning the hyper-reduction only would amount to learn $\Phi^t \mathbf{F}(\Phi\hat{\boldsymbol{\rho}})$ instead, where $\hat{\boldsymbol{\rho}}$ is the compression of the full order trajectories.

Remark 9. *To simplify the notations in what follows, we will always consider a constant time step Δt such that, for each integer n, we write $t_n = (n-1)\Delta t$. Moreover, we write $\boldsymbol{\rho}^n_{\boldsymbol{\mu}} := \boldsymbol{\rho}(t_n;\boldsymbol{\mu})$ for all $(n, \boldsymbol{\mu})$.*

In everything that follows, we consider an Euler explicit time integration of Equation 5.9:

$$\begin{cases} \hat{\boldsymbol{\rho}}^{n+1}_{\boldsymbol{\mu}} = \hat{\boldsymbol{\rho}}^n_{\boldsymbol{\mu}} + \Delta t\, \hat{F}_\theta(\hat{\boldsymbol{\rho}}^n_{\boldsymbol{\mu}}, \boldsymbol{\mu}), \\ \hat{\boldsymbol{\rho}}^1_{\boldsymbol{\mu}} = \hat{\boldsymbol{\rho}}_{init} \in \mathbb{R}^{N_m}, \end{cases} \tag{5.10}$$

i.e. the neural network's objective is to calculate a time integrator for a single iteration.

We will compare two approaches that produce the same type of network, but differ in their training strategies. In either case, the network will be a multi-linear perceptron (MLP), with hyper-parameters described in detail below.

The dataset is assembled as follows:

- **Snapshot construction:** For each $\boldsymbol{\mu} \in \mathcal{P}_\mu$ we solve the high-fidelity Equation 5.1 from which we save the pairs
$$\left\{ \left(\boldsymbol{\rho}_{\boldsymbol{\mu}}, \mathbf{F}\left(\boldsymbol{\rho}_{\boldsymbol{\mu}};\boldsymbol{\mu}\right) \right);\ \boldsymbol{\mu} \in \mathcal{P}_\mu \right\}. \tag{5.11}$$
- **Construction of the reduced space:** An SVD is performed on the snapshots to form and store the encoding matrix Φ^T as well as the projections of the states and the fluxes
$$\hat{\boldsymbol{\rho}}^{proj}_{\boldsymbol{\mu}} := \Phi^T \boldsymbol{\rho}_{\boldsymbol{\mu}}, \quad \hat{\mathbf{F}}^{proj}_{\boldsymbol{\mu}} := \Phi^T \mathbf{F}\left(\boldsymbol{\rho}_{\boldsymbol{\mu}};\boldsymbol{\mu}\right). \tag{5.12}$$
- **Neural network I/O:** The network will therefore have inputs $\left(\hat{\boldsymbol{\rho}}^{proj}_{\boldsymbol{\mu}}, \boldsymbol{\mu}\right)$ to predict the targets $\hat{\mathbf{F}}^{proj}_{\boldsymbol{\mu}}$.

Our neural network is specifically designed to acquire knowledge about an integration flux, which in other words means it grasps the underlying physics of our problem or the model. During training, we employ two well-established methods commonly used in this field: model fitting and trajectory fitting [1]. The former involves straightforward computation based on the neural network's output, while the latter is conceptually more intuitive as it remains independent of the model and directly compares solutions.

2.1 Neural closure (NC):

Our first approach of the deep-hyper-reduction strategy is to compare directly the output of the neural network to its prediction target, which leads to consider the minimization of the model error:

$$\mathcal{L}^{NC}(\theta) = \mathrm{MSE}\left(\hat{F}^{NC}_\theta(\hat{\boldsymbol{\rho}}^{proj}_{\boldsymbol{\mu}};\boldsymbol{\mu}), \hat{\mathbf{F}}^{proj}_{\boldsymbol{\mu}}\right) = \frac{1}{N_\mu N_t} \sum_{\substack{\boldsymbol{\mu}\in\mathcal{P}_\mu \\ t_n\in\mathcal{P}_t}} \left\| \hat{F}^{NC}_\theta(\hat{\boldsymbol{\rho}}^{proj}_{\boldsymbol{\mu}}(t_n);\boldsymbol{\mu}) - \hat{\mathbf{F}}^{proj}_{\boldsymbol{\mu}}(t_n) \right\|^2. \tag{5.13}$$

2.2 Differentiable programming (DP):

In the second approach, we attempt to learn the whole integrator over a given number $K \in \mathbb{N}$ of time steps. More precisely, writing the explicit time discretization of Equation 5.9 as:

$$\hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{n+1} = \hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{n} + \hat{F}_{\theta}^{DP}(\hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{n}; \mu)\Delta t =: S_{\theta}^{1}(\hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{n}), \tag{5.14}$$

we define by induction the integrator S_{θ}^{K} yielding the reduced state $\hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{n+K}$:

$$\begin{aligned}
\hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{n+2} &= \hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{n+1} + \hat{F}_{\theta}^{DP}(\hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{n+1}, \boldsymbol{\mu})\Delta t \\
&= S_{\theta}^{1}(\hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{n}) + \hat{F}_{\theta}^{DP}(S_{\theta}^{1}(\hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{n}), \boldsymbol{\mu})\Delta t \\
&=: S_{\theta}^{2}(\hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{n}), \\
\hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{n+3} &= \cdots, \\
\hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{n+K} &=: S_{\theta}^{K}(\hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{n}).
\end{aligned}$$

We will hereafter denote the reduced state at time t_{n+k} predicted by the neural network from a low-dimension state $\hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{n}$ at time t_n by:

$$\hat{\boldsymbol{\rho}}_{\boldsymbol{\mu},\theta}^{DP,n+k} := S_{\theta}^{k}(\hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{n}). \tag{5.15}$$

Our baseline will be the compression of the high-fidelity state $\hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{proj} := \Phi^T \boldsymbol{\rho}_{\boldsymbol{\mu}}$, thus we search for optimal parameters θ_{opt} such that:

$$S_{\theta_{opt}}^{K}(\hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{proj,n}) = \hat{\boldsymbol{\rho}}_{\boldsymbol{\mu},\theta_{opt}}^{DP,n+K} \approx \hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{proj,n+K}. \tag{5.16}$$

We therefore minimize the following loss:

$$\mathcal{L}_{K}^{DP}(\theta) = \mathrm{MSE}(\hat{\boldsymbol{\rho}}_{\boldsymbol{\mu},\theta}^{DP}, \hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{proj}) = \frac{1}{N_\mu}\frac{1}{N_t - K} \sum_{\substack{1 \le n \le N-K \\ \boldsymbol{\mu} \in \mathcal{P}_\mu}} \left\| \hat{\boldsymbol{\rho}}_{\boldsymbol{\mu},\theta}^{DP,n+K} - \hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{proj,n+K} \right\|^2. \tag{5.17}$$

Another loss that we will consider is the time average of the previous one:

$$\mathcal{L}_{K,int}^{DP}(\theta) = \frac{1}{N_\mu}\frac{1}{N_t - K} \sum_{\substack{1 \le n \le N-K \\ \boldsymbol{\mu} \in \mathcal{P}_\mu}} \left(\frac{1}{K} \sum_{1 \le k \le K} \left\| \hat{\boldsymbol{\rho}}_{\boldsymbol{\mu},\theta}^{DP,n+k} - \hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{proj,n+K} \right\|^2 \right). \tag{5.18}$$

This formulation has the advantage of using information not only from $\hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{n+K}$ but also from the entire trajectory $\{\hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{n+k}, 0 \le k \le K\}$. In this way, we hope to obtain a more generalizable neural network.

We observe that for $K = 1$, this approach boils down to the model learning introduced before, indeed:

$$\begin{aligned}
\hat{\boldsymbol{\rho}}_{\boldsymbol{\mu},\theta}^{DP,n+1} - \hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{proj,n+1} &= \hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{n} + \hat{F}_{\theta}^{DP}(\hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{n}; \mu)\Delta t - \Phi^T \left(\boldsymbol{\rho}_{\boldsymbol{\mu}}^{n} + \mathbf{F}(\boldsymbol{\rho}_{\boldsymbol{\mu}})\Delta t \right) && (5.19) \\
&= (\hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{n} - \hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{n}) + \left(\hat{F}_{\theta}^{DP}(\hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{n}; \mu) - \hat{\mathbf{F}}_{\boldsymbol{\mu}}^{proj} \right) \Delta t, && (5.20) \\
&= \left(\hat{F}_{\theta}^{DP}(\hat{\boldsymbol{\rho}}_{\boldsymbol{\mu}}^{n}; \mu) - \hat{\mathbf{F}}_{\boldsymbol{\mu}}^{proj} \right) \Delta t, && (5.21)
\end{aligned}$$

with the difference that the flow discrepancy is now rescaled by the time step. This form of renormalization is not trivial, as it's widely recognized that neural networks tend to benefit from techniques such as batch

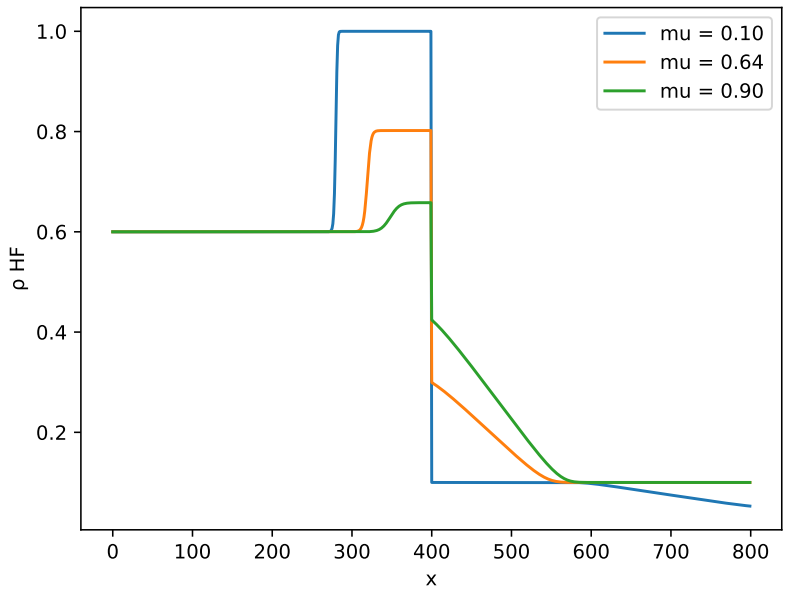


Figure 5.4: Vehicle densities at final time for different values of maximum density on the second road.

normalization [86]. We will therefore conduct a comparison between model training without this renormalization and trajectory learning specifically for the case when K equals 1. This comparison will help emphasize the impact of time rescaling.

3 Numerical results

3.1 Test-case presentation

We want to describe different types of dynamics in the two-road model presented in Figure 1.2, which are influenced by the maximum density allowed on each road and denoted by $\boldsymbol{\mu} := \boldsymbol{\rho}^{max} \in \mathbb{R}^2$. We initialize the road network with constant values that are respectively 0.6 on the road going in the junction and 0.1 for the outgoing one. We impose constant density of 0.6 at the network entrance and a free exit condition. Setting $\mu_1 = 1$, and varying $\mu_2 = \rho_2^{max}$ between 0.1 and 0.9 results in different congestion phenomena, as shown in Figure 5.4. Low values of μ_2 result in the most congested intersection, with a large shock wave propagating towards the left. Conversely, higher values of μ_2 lead to smaller shock waves, as vehicles are permitted to cross the intersection, resulting in a rarefaction wave propagating in the right direction.

Symbol	Name	Values
T	Final time	0.7
N_t	Number of time steps	562
N_x	Total number of cells on the network	800
N_m	Number of modes	$\{5, 20\}$
N_μ	Number of values taken by $\boldsymbol{\mu}$	300
$\boldsymbol{\mu}$	Model parameter ($\boldsymbol{\rho}^{\max}$)	$\{1\} \times [0.1, 0.9]^{N_\mu}$

Table 5.1: Full and reduced model parameters.

Architecture and hyperparameters: All approaches are trained on the same fully connected neural network of 4 layers with 32 neurons each and $\tanh$ activation functions. We trained the networks with Adam optimizer, batches of size 16 and a 30% dropout rate. The learning rate was scheduled as follows:

$$\textbf{iterations} \quad \textbf{1-10}: \quad lr = 10^{-3},$$
$$\textbf{iterations} \quad \textbf{k>10}: \quad lr_k = lr_{k-1} \times e^{-0.1}.$$

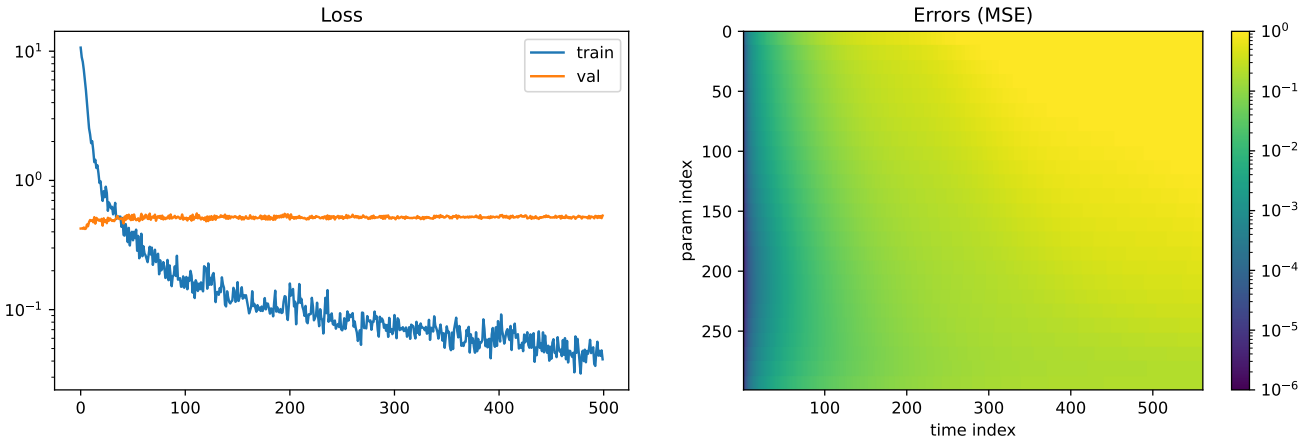
We uniformly sampled 9 values of μ_2 between 0.1 and 0.9 as validation data for both approaches.

3.2 Results

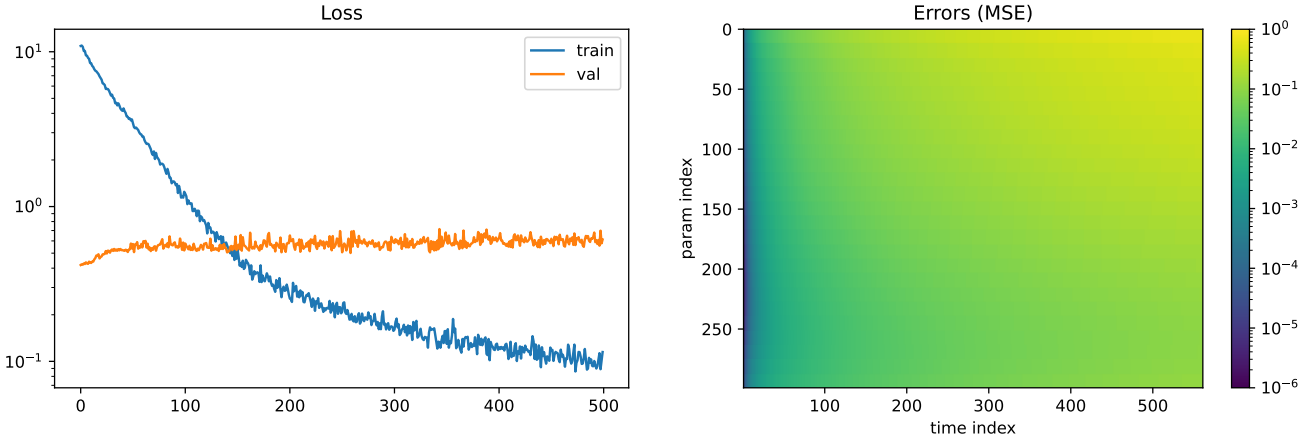
We base our comparisons on the DEIM method. Figure 5.2 shows the limitations of this approach for our problem, since the state is not well captured at all with oscillations instead of shock, and no useful information about its height. The DEIM state is essentially equal to the initial condition.

Training: In Figure 5.5, Figure 5.6, and Figure 5.7, the left side displays learning processes for NC and DP approaches with 5 and 20 modes, both with and without time averaging. On the right side, the figures depict the corresponding MSE errors at various time points, calculated for 12 equidistant values of μ_2 within the range of [0.1, 0.9]. We observe that even though the training loss is always decreasing, the validation loss is constant for NC. We thus stop the training early, after 500 epochs. However, the validation losses for DP are decreasing with the training, hence we continue the learning for 2000 epochs. It seems from the MSEs that the worst predictions are generally made for long times and small parameters, i.e. the prediction of the final times of shock wave propagation. The higher values of μ_2 that correspond to rarefactions are, conversely, more successfully learned. But, except for perhaps the NC MSE, the MSE is neither linear nor convex. This highlights the nonlinearity of the effect of the parameters on the predictions.

Predictions: Figure 5.8 shows the comparison of NC and DP with 5 modes and without time averaging. It displays time errors on the right side, and corresponding states at time iteration 400 on the left side. The DEIM modes are unsurprisingly distant from the references, resulting in incorrect density. The NC approach enhances the outcome by providing a reasonable shock height with fewer oscillations. However, the location of the front is poorly anticipated by a wavefront positioned approximately halfway from the reference wavefront. In contrast, the DP method shows great reconstruction properties since the modes are well learned, allowing



(a) 5 modes



(b) 20 modes

Figure 5.5: Loss convergence over 500 epochs for the Neural Closure strategy.

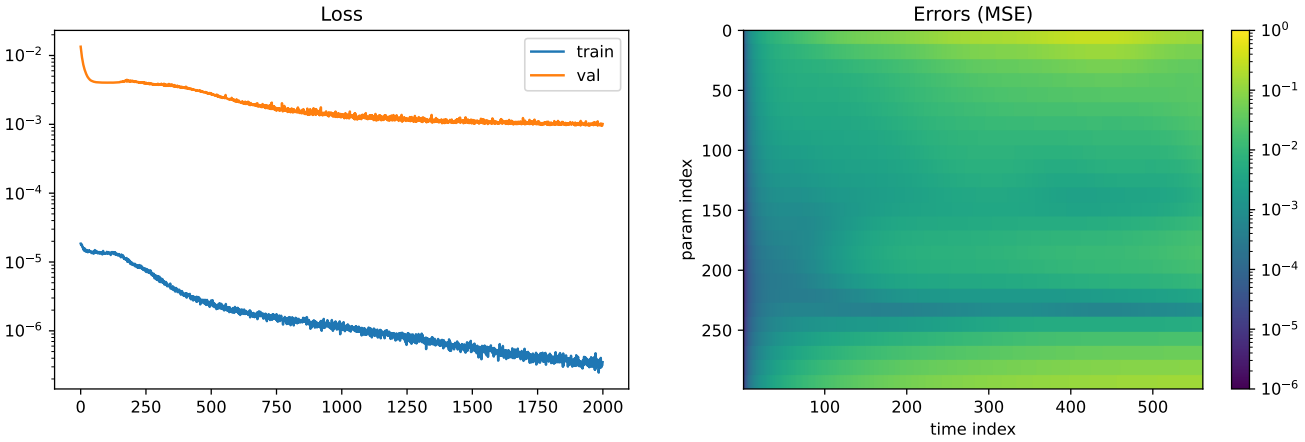
accurate reconstruction of the full state. With only 5 modes, the FV state is somewhat diffuse, but increasing the number of modes allows the FV wavefront to be tracked very well, with a few spurious oscillations due to modal decomposition.

The same comparison with the 20 modes is made in Figure 5.9. We see more oscillations in the reduced solution of reference, and a lot of small oscillations for the DEIM state. The NC yields a too small amplitude and the same wavefront as before, whereas the DP's amplitude is truly above the reference but still with a reasonable wavefront location.

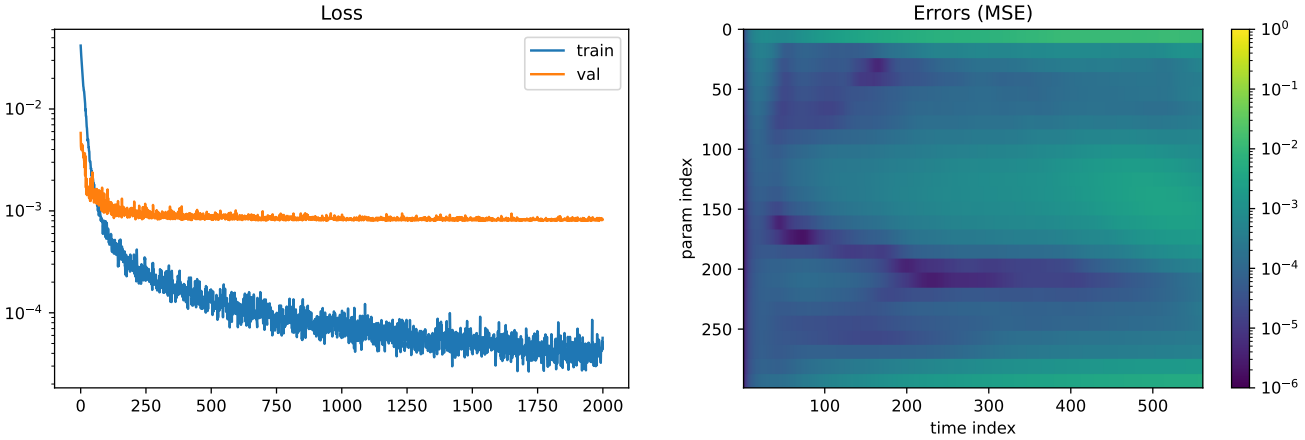
Figure 5.10 and Figure 5.11, demonstrate that successively adding time iterations ($K = 50$) and time averaging greatly improves the results. Indeed, the full solution is very well fitted both with respect to amplitude and wavefront locations. Furthermore, the error in time is flattened, suggesting a better stability.

4 Conclusion and perspectives

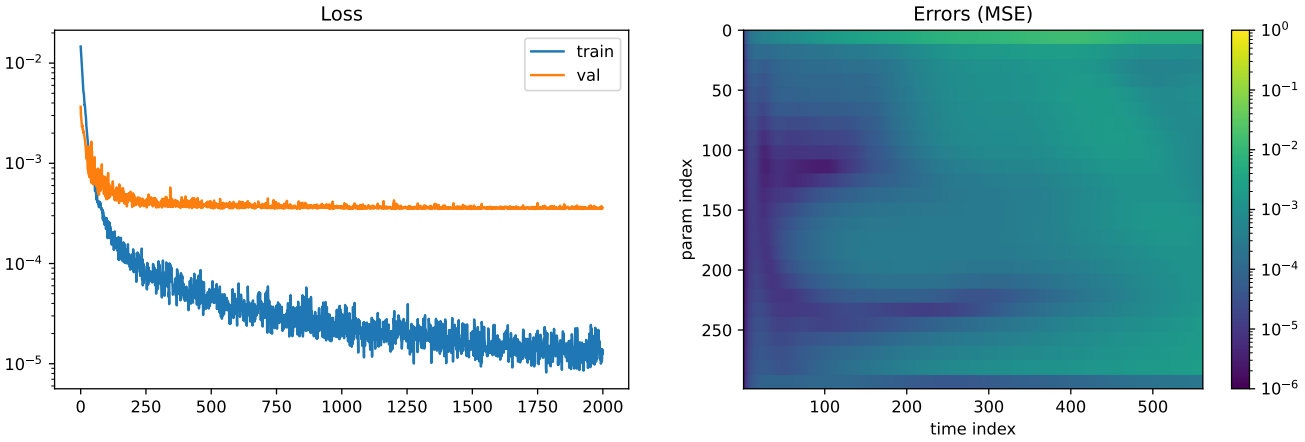
In this work, we have introduced a new way to compute a closed hyper-reduction based on a deep learning approach. Two methods are investigated, inspired from the trajectory fitting and the model fitting frameworks. The numerical results obtained on the single-junction network have shown that the trajectory fitting method seems effective in providing an accurate reduced dynamics for all kind of flows, even the most congested ones.



(a) 5 modes 1 iteration

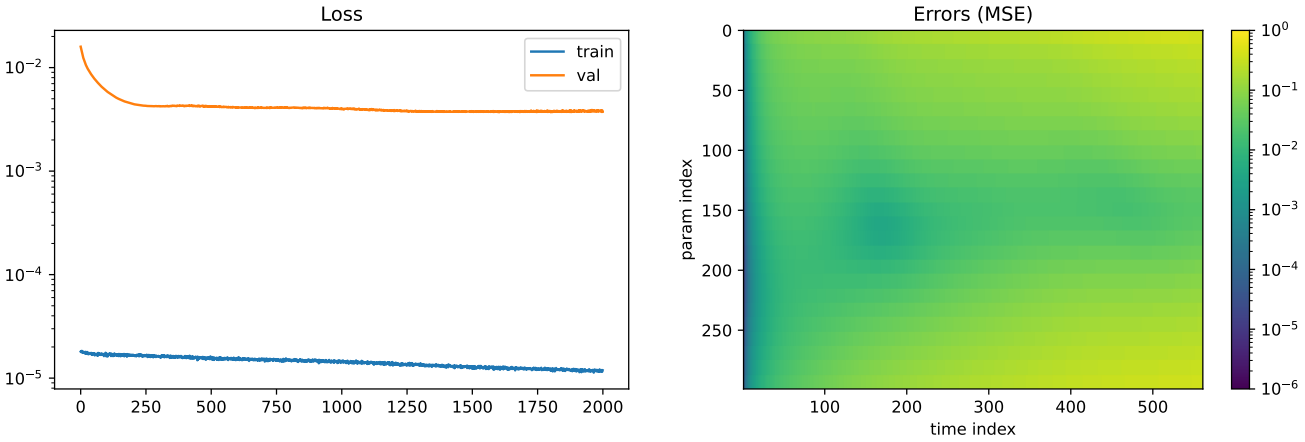


(b) 5 modes 50 iterations

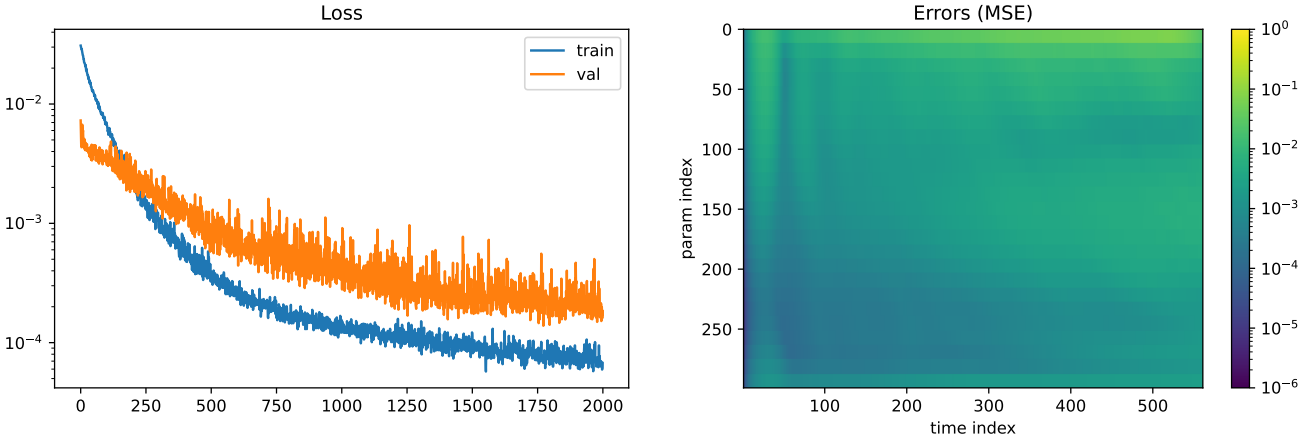


(c) 5 modes 50 iterations with time integration

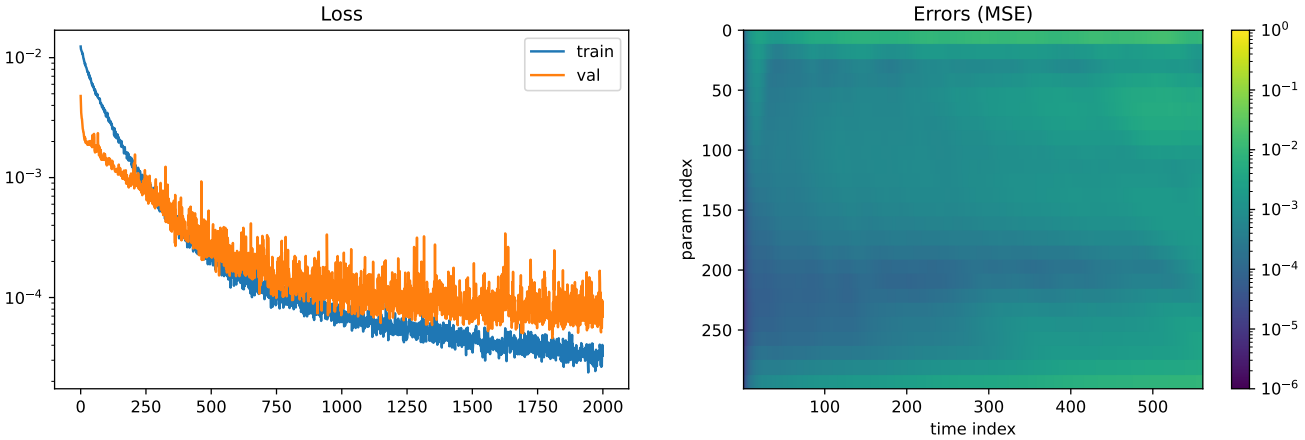
Figure 5.6: Loss convergence over 2000 epochs for the Differentiable Physics strategy using 5 modes.



(a) 20 modes 1 iteration



(b) 20 modes 50 iterations



(c) 20 modes 50 iterations with time integration

Figure 5.7: Loss convergence over 2000 epochs for the Differentiable Physics strategy using 20 modes.

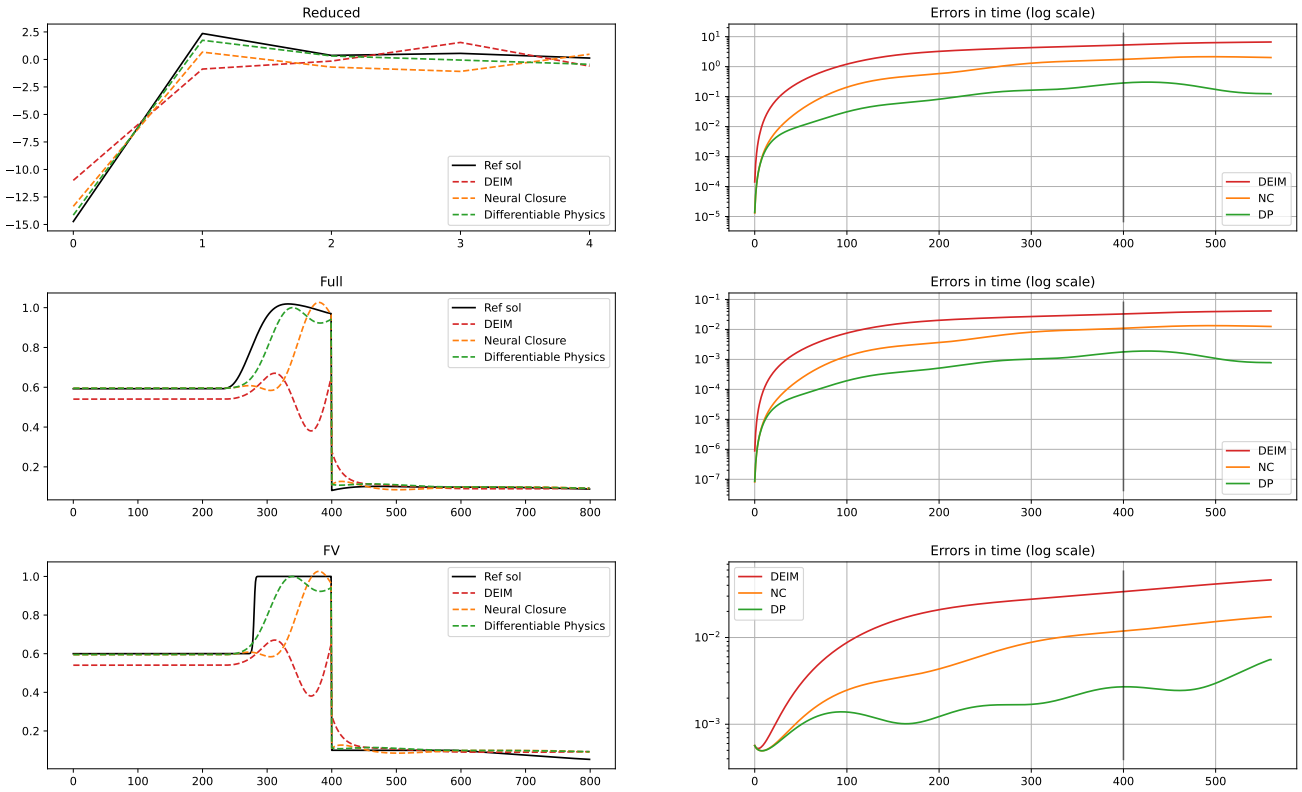


Figure 5.8: Compare 5 modes 1 iter

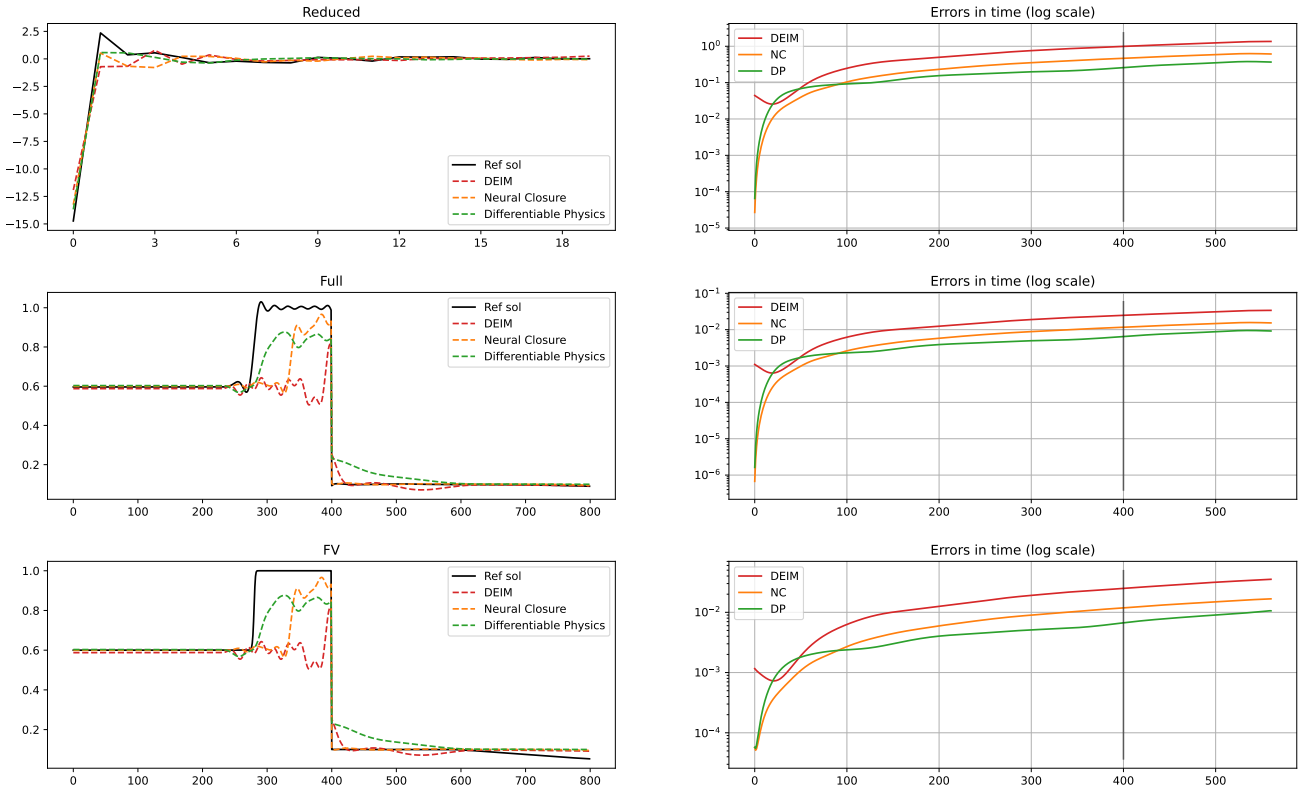


Figure 5.9: Compare 20 modes 1 iter

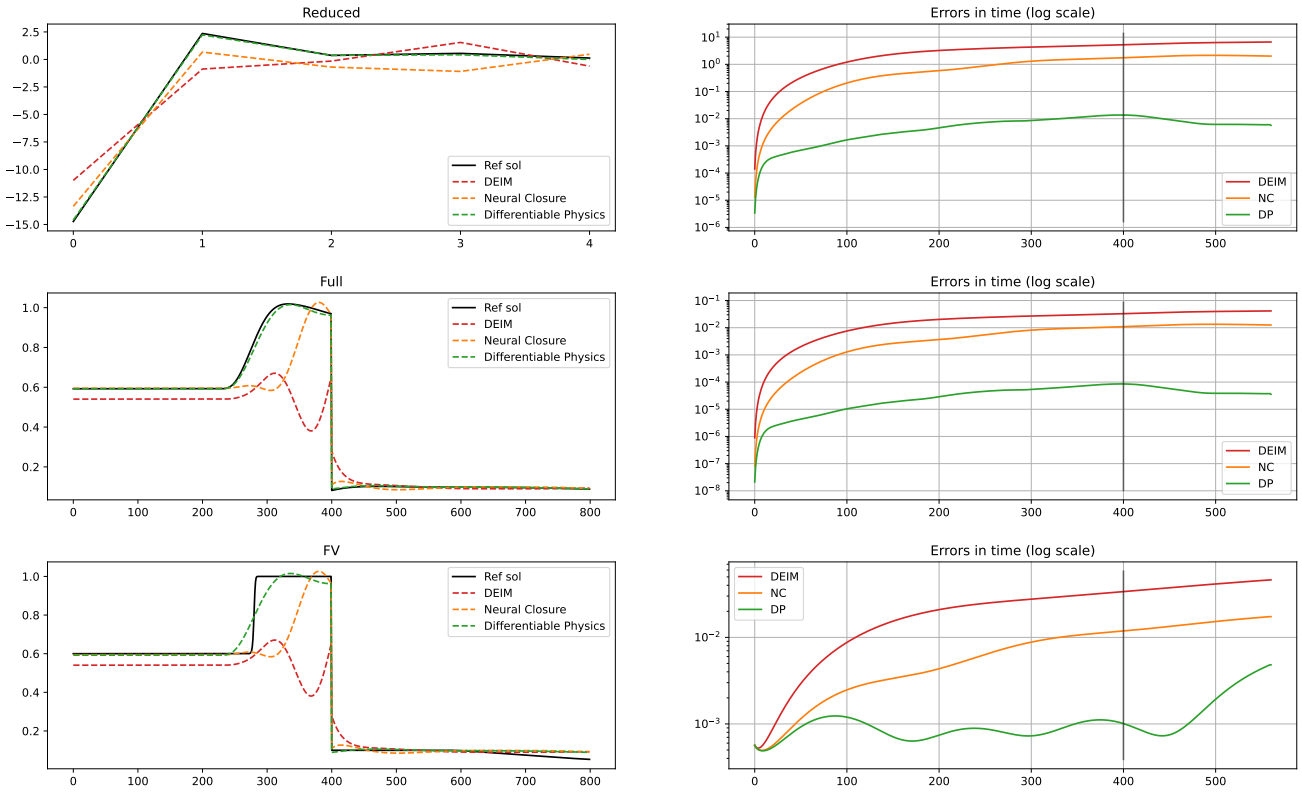


Figure 5.10: Compare 5 modes 50 iter with time integration

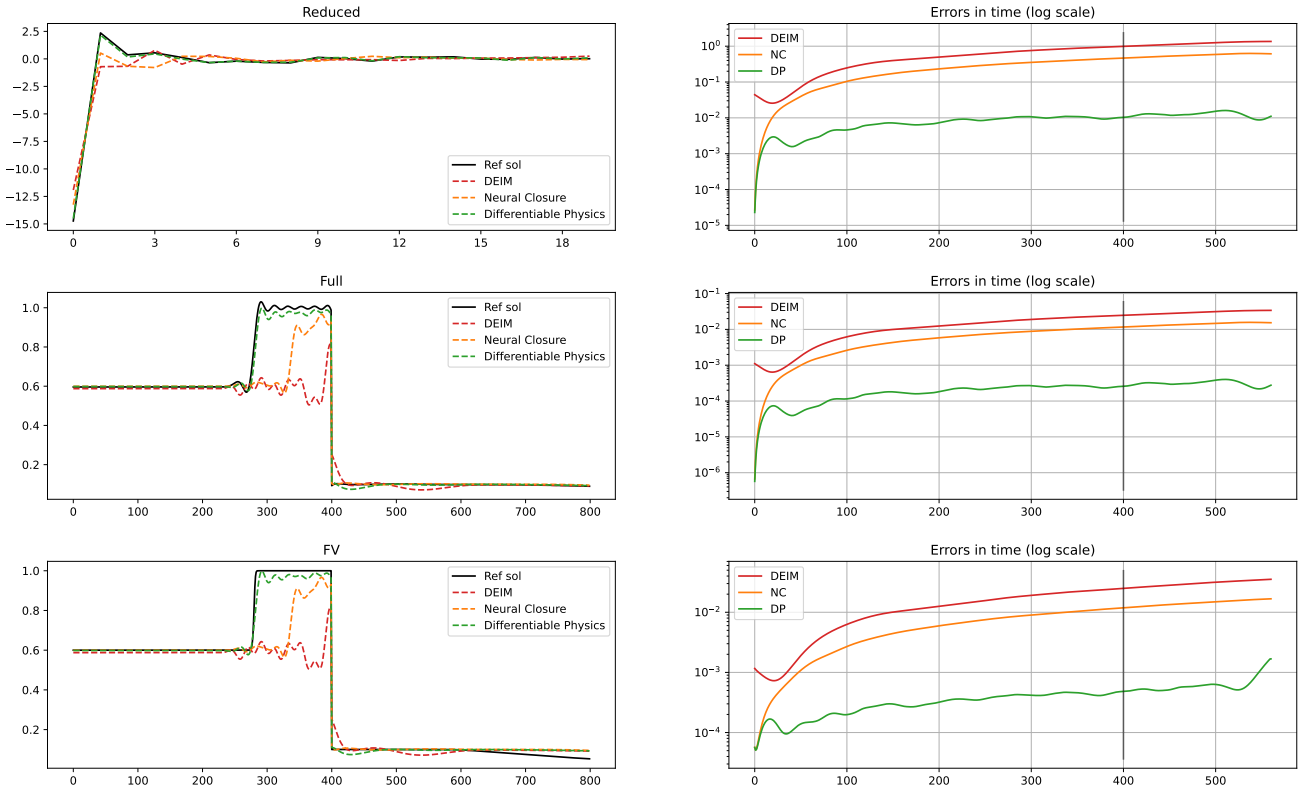


Figure 5.11: Compare 20 modes 50 iter with time integration

This work is still preliminary and can be pursued in several directions. Firstly, it would be interesting to extend the method for more complex networks as those introduced in chapter 2. A suitable interface to the optimal control algorithm would enable the problem to be extended to larger networks, which could lead to real applications using standard computers, and therefore more easily deployable.

Secondly, the training process can be improved with better architecture and/or parameters. It would be interesting to use reccurent neural networks or long-short-term memory, as they are suited for time evolving problems. Investigations on suitable activation functions and normalizations should help improve the overall process.

It would also be interesting to test numerically the conjecture made by Noack et al. [70] and mentioned in [1] that the network trained with model regression (Neural Closure) extrapolates better outside the training interval than the one trained with trajectory regression (Differentiable Physics). However, the latter is claimed to be more accurate in the reconstructive regime, i.e. inside the training interval.

Chapter 6

Numerical schemes for mixture theory models with filling constraint: application to biofilm ecosystems.

This chapter is a proceedings written during the CEMRACS summer school in 2022 in collaboration with Leo Meyers and Florent Noisette under the supervision of Bastien Polizzi, Sébastian Minjeaud, Olivier Bernard and Thierry Goudon. It is currently being reviewed for the journal ESAIM: Proceedings and Surveys.

1 Introduction

There are many physical cases of flows composed of different gas or liquids interacting together. For example, tissue bodies and tumors can be described as a set of interacting viscoelastic materials. Powder-snow avalanches can be described as a mixture of fluid phases. Similarly, the rheology of the gut microbiota and its interactions with chyme (a mixture of partially digested food and water) and the host can be modeled using mixture theory [57]. Complex flows can also be found in many engineering applications involving multiphase systems such as boiling water in nuclear reactors. Therefore, the framework of mixture theory is a common tool to model and study complex flows.

Mathematical models based on mixture theory take the form of systems of partial derivative equations, coupled with algebraic constraints. The theoretical analysis of such systems and the characterization of the qualitative properties of the solutions are extremely complicated [9, 45, 52, 88]. Thus, it is important to develop efficient numerical methods able to accurately capture the solutions [15, 21, 22, 34].

In this article, we are interested in applying mixture models to describe biofilm dynamics. Indeed, mixture theory revealed a powerful approach to represent microbial biofilms where a consortium of cells is embedded in a polymeric structure [22, 74, 75].

In mixture theory, the unknowns of the model are requested to satisfy certain constraints. As far as the continuous equations are considered, several equivalent formulations of these constraints can be derived and used to bring out the properties of the model. However, the preservation of these constraints by a numerical scheme is a challenge and, once a discretization setup has been adopted, it is not clear that all the formulations of the constraints remain equivalent. This issue can induce a loss of stability and accuracy, and eventually a

dramatic loss of key physical properties of the simulated flows. Thus, we adapt and extend the numerical scheme proposed in [10] in order to preserve these constraints. The numerical scheme will be tested and illustrated with a multiphasic model representing the development of a photosynthetic biofilm, with the application for biofuel, protein, or drug production.

The paper is organized as follows. The first section is dedicated to the mixture theory framework with a presentation of the simplified model used to test our numerical scheme. The second section details the numerical scheme and its properties. The third section presents the results and comparison with standard numerical schemes.

2 Mixture theory framework: application to biofilms

2.1 Mixture theory framework

The mixture theory framework [80], also known as mixture mechanics or continuum mechanics for fluid dynamics, enables describing multi-phasic systems at the mesoscopic scale which is an intermediary scale between microscopic and macroscopic scales. It was introduced in the 1960s by Truesdell [91, 93, 92] and generalizes Navier-Stokes equations to multi-phasic systems. The mixture theory framework assumes that each component of the mixture might be present at every point in space and at any time. Moreover, the system's physical properties (ex. viscosity, incompressibility, ...) are naturally included.

Consider a mixture of k components indexed by $\alpha \in [\![1, k]\!]$. Each component is locally described by its volumetric mass density ρ_α, its volume fraction ϕ_α, and its local velocity v_α. The volume fraction represents the relative volume occupied by a component in an elementary normalized piece of volume. Thus, assuming that there is no vacuum they satisfy the algebraic constraint

$$\sum_{\alpha=1}^{k} \phi_\alpha = 1. \tag{6.1}$$

The mixture dynamic depends on mass transfers which are modeled through mass balance equations (6.2a) and the local forces applied to the system which are accounted for through momentum balance equations (6.2b). Thus, for each component the state variables satisfy the equations:

$$\partial_t(\rho_\alpha\phi_\alpha) + \nabla_x \cdot (\rho_\alpha\phi_\alpha v_\alpha) = \Gamma_\alpha, \tag{6.2a}$$

$$\partial_t(\rho_\alpha\phi_\alpha v_\alpha) + \nabla_x \cdot (\rho_\alpha\phi_\alpha v_\alpha \otimes v_\alpha) + \nabla_x \pi_\alpha + \phi_\alpha \nabla_x P = \nabla_x \cdot (\phi_\alpha \tau_\alpha) + F_\alpha + \phi_\alpha\rho_\alpha \boldsymbol{g} + \Gamma_\alpha v_\alpha, \tag{6.2b}$$

where Γ_i is the mass exchange term, π_α is the elastic tensor, P is the common pressure, τ_α the viscous stress tensor, F_α the friction forces, and $\boldsymbol{g}$ the gravity force. Depending on the considered application some forces can be neglected and some others might be added.

Depending on the targeted application one can add for each component an extra equation for the evolution of the density ρ_α. Nevertheless, liquids are weakly compressible, especially when pressure variations are small. Therefore, in most cases, for liquids the component densities ρ_α can be assumed constant. When all the component volumetric mass densities are assumed constant, the mass balance equations (6.2a) are equivalent to $\partial_t(\phi_\alpha) + \nabla_x \cdot (\phi_\alpha v_\alpha) = \Gamma_\alpha/\rho_\alpha$. Then summing these equations for each phase leads to the pseudo

incompressibility constraint:

$$\nabla_x \cdot \left(\sum_\alpha \phi_\alpha v_\alpha \right) = \sum_\alpha \frac{\Gamma_\alpha}{\rho_\alpha}. \tag{6.3}$$

This means that the local divergence of the averaged mixture velocity is equal to the local volume variation induced by mass exchanges.

The elastic tensor π_α can be interpreted as the internal pressure of the component. There are several ways to model this term depending on the nature of the component. When the component α represents particles, as in [10], there is a close-packing limit. This property can be enforced by using an appropriate expression for π_α as

$$\pi_\alpha = \gamma_\alpha \frac{\phi_\alpha^{\beta_\alpha}}{\phi_\alpha^\star - \phi_\alpha}, \quad \text{with} \quad \gamma_\alpha > 0, \quad \text{and} \quad \beta_\alpha > 1, \tag{6.4}$$

where $0 < \phi_\alpha^\star < 1$ is the so-called close-packing volume fraction limit. When the component α represents softer material like living tissues it can take the form of standard pressure law:

$$\pi_\alpha = \gamma_\alpha \left(\frac{\phi_\alpha}{\phi_\alpha^\star} \right)^{\beta_\alpha}, \quad \text{with} \quad \gamma_\alpha > 0, \quad \text{and} \quad \beta_\alpha \geqslant 1, \tag{6.5}$$

where $0 < \phi_\alpha^\star < 1$ is a threshold, see [22, 76]. More complex laws, based on the Flory–Huggins theory:

$$\pi_\alpha = -\gamma_\alpha \big(\ln(1 - \phi_\alpha) + \phi_\alpha + \phi_\alpha^2 \big), \quad \text{with} \quad \gamma_\alpha > 0, \tag{6.6}$$

enable accounting for colligative properties at low concentrations, see [24].

The viscous stress tensor τ_α is defined by

$$\tau_\alpha = \mu_\alpha \phi_\alpha \big(\nabla v_\alpha + {}^t\nabla v_\alpha - \tfrac{2}{3} (\nabla \cdot v_\alpha) \mathrm{Id} \big), \tag{6.7}$$

where the constant $\mu > 0$ stands for the component dynamic viscosity and ${}^t\nabla v_\alpha$ stands for the transpose of the velocity differential matrix.

The friction force F_α is induced by the difference in the relative speed of the mixture components:

$$F_\alpha = \sum_{\alpha' \neq \alpha} f_{\alpha,\alpha'} (v_{\alpha'} - v_\alpha) \tag{6.8}$$

with $f_{\alpha,\alpha'}$ the friction force law between the components pair α and α'. As a first approximation, it can be assumed that $f_{\alpha,\alpha'}$ is a strictly positive constant. However, the friction between two components should vanish when one of them disappears. Thus, a more realistic alternative is to consider that friction depends on the local composition and use instead $f_{\alpha,\alpha'} (\phi_\alpha \phi_{\alpha'})^{r_{\alpha,\alpha'}}$. Nevertheless, the total momentum conservation principle enforces that

$$\sum_\alpha F_\alpha = 0.$$

Dissolved components, like substrate, can be included. A dissolved component p within a phase α is described through its concentration θ_p. In addition to the transport by the phase, it can also diffuse within the phase at a

rate D_p. Thus, the mass balance equations for a dissolved component within the phase α writes:

$$\partial_t(\rho_\alpha \phi_\alpha \theta_p) + \nabla_x \cdot (\rho_\alpha \phi_\alpha \theta_p v_\alpha) - \nabla_x \cdot (\rho_\alpha \phi_\alpha D_p \nabla_x \theta_p) = \Gamma_p. \tag{6.9}$$

where again the source term Γ_p represents the mass exchange associated to component p.

2.2 Mixture model for biofilm

We focus on a simplified 1D model for biofilms. Biofilms are made of microorganisms $\mathcal{A}$ (microalgae, bacteria, or a consortium of both) and an extra-cellular matrix $\mathcal{E}$. The biofilm is usually immersed in water $\mathcal{L}$. Therefore, according to mixture theory framework, see section 2.1, each component $\alpha \in \{\mathcal{A}, \mathcal{E}, \mathcal{L}\}$ is described through three macroscopic variables: the mass density ρ_α, the volume fraction ϕ_α, and the velocity v_α. By definition, the volume fractions satisfy at any time the algebraic volume-filling constraint (6.1) which reads in this case: $\phi_\mathcal{A} + \phi_\mathcal{E} + \phi_\mathcal{L} = 1$. In the one-dimensional case, the mass balance equations (6.2a) writes:

$$\partial_t(\rho_\alpha \phi_\alpha) + \partial_x(\rho_\alpha \phi_\alpha v_\alpha) = \Gamma_\alpha, \qquad \alpha \in \{\mathcal{A}, \mathcal{E}, \mathcal{L}\}. \tag{6.10}$$

In this context, the volumetric mass densities ρ_α can be assumed to be constant. Thus, the mixture averaged velocity satisfies the pseudo incompressibility constraint (6.11) which writes here:

$$\partial_x\big(\phi_\mathcal{A} v_\mathcal{A} + \phi_\mathcal{E} v_\mathcal{E} + \phi_\mathcal{L} v_\mathcal{L}\big) = \frac{\Gamma_\mathcal{A}}{\rho_\mathcal{A}} + \frac{\Gamma_\mathcal{E}}{\rho_\mathcal{E}} + \frac{\Gamma_\mathcal{L}}{\rho_\mathcal{L}}. \tag{6.11}$$

For biofilms, there are various biological processes to be taken into account. The main processes are growth, extra-cellular matrix excretion, and death. These reactions are schematically represented in Table 6.1. The parameters η_α are pseudo-stoichiometric coefficients that quantify how much a reactant (ex. liquid, algae, substrate, ...) or a product (ex. algae, extra-cellular matrix, ...) is consumed or produced when a reaction occurs. The functions ψ_i are the reaction rates. They describe the speed at which reactions take place as a function of the local composition of the mixture. The source terms read as follows:

$$\Gamma_\mathcal{A} = \psi_g - \psi_e - \psi_d, \qquad \Gamma_\mathcal{E} = \psi_e + \eta_\mathcal{E}\psi_d, \qquad \Gamma_\mathcal{L} = (1-\eta_\mathcal{E})\psi_d - \eta_\mathcal{L}\psi_g.$$

	Biological reaction representation		
Name	Reactant(s)	Rate	Product(s)
Growth	$\eta_\mathcal{L}\mathcal{L} + \eta_\mathcal{S}\mathcal{S}$	$\xrightarrow{\psi_g}$	$\mathcal{A}$
Excretion	$\mathcal{A}$	$\xrightarrow{\psi_e}$	$\mathcal{E}$
Death	$\mathcal{A}$	$\xrightarrow{\psi_d}$	$\eta_\mathcal{E}\mathcal{E} + (1-\eta_\mathcal{E})\mathcal{L}$

Table 6.1: Schematic representation of the biochemical reactions considered in the model.

The growth is mainly induced by substrate ($\mathcal{S}$) assimilation and liquid ($\mathcal{L}$) absorption. However, as a first approximation, we assume that the substrate is in excess. Thus, the growth rate ψ_g takes the form $\psi_g = \mu_g \rho_\mathcal{A} \phi_\mathcal{A} \phi_\mathcal{L}$, where μ_g is the maximal growth rate. The extra-cellular matrix excretion ψ_e and the death rate ψ_d are assumed to be proportional to the quantity of microalgae, thus $\psi_e = \mu_e \rho_\mathcal{A} \phi_\mathcal{A}$ and $\psi_d = \mu_d \rho_\mathcal{A} \phi_\mathcal{A}$ respectively.

Nevertheless, biofilms are very complex ecosystems and the biological processes are very simplified here. Thus, a model extension accounting for substrate and oxygen is presented in section 5.3.

In the one-dimensional case and neglecting the gravity, for $\alpha \in \{\mathcal{A}, \mathcal{E}, \mathcal{L}\}$ the momentum balance equations simplify into:

$$\partial_t(\rho_\alpha \phi_\alpha v_\alpha) + \partial_x\big(\rho_\alpha \phi_\alpha v_\alpha^2\big) + \partial_x \pi_\alpha = -\phi_\alpha \partial_x P + \frac{4}{3}\partial_x(\mu_\alpha \phi_\alpha \partial_x v_\alpha) + F_\alpha + \Gamma_\alpha v_\alpha.$$

To keep the model as simple as possible, let us assume that the elastic tensor takes the form of a pressure law, see equation (6.5), for the tissues (ie. algae and extra-cellular matrix). Since the liquid phase is not elastic this term is null for the liquid, namely $\pi_\mathcal{L} = 0$. Similarly, let us assume that the friction forces are constant and symmetric. Thus, in the expression (6.8) for F_α, the term $f_{\alpha,\alpha'}$ for $(\alpha, \alpha') \in \{\mathcal{A}, \mathcal{E}, \mathcal{L}\}^2$ and $\alpha \neq \alpha'$ are constant and such that $f_{\alpha,\alpha'} = f_{\alpha',\alpha}$.

The model is supplemented by boundary conditions. Let $\Omega = [0, L]$ be the domain and $\partial\Omega$ its boundary. In 1D, the domain should correspond to a biofilm core drilling in the orthogonal axis of the support where the biofilm develops. The velocities at the bottom of the domain, which corresponds to the surface on which the biofilm develops, vanish $v_\alpha(0) = 0,\ \alpha \in \{\mathcal{A}, \mathcal{E}, \mathcal{L}\}$. However, the velocity on the top must satisfy a constraint induced by the incompressibility constraint (6.3). Indeed, the integration over the whole domain of equation (6.3) combined with the null velocity at the bottom leads to

$$\big(\phi_\mathcal{A} v_\mathcal{A} + \phi_\mathcal{E} v_\mathcal{E} + \phi_\mathcal{L} v_\mathcal{L}\big)(x = L) = \int_0^L \left(\frac{\Gamma_\mathcal{A}}{\rho_\mathcal{A}} + \frac{\Gamma_\mathcal{E}}{\rho_\mathcal{E}} + \frac{\Gamma_\mathcal{L}}{\rho_\mathcal{L}}\right) dx$$

To enforce this condition, let assume that on the top, the velocities are given by $v_{x=L} = \int_0^L \left(\frac{\Gamma_\mathcal{A}}{\rho_\mathcal{A}} + \frac{\Gamma_\mathcal{E}}{\rho_\mathcal{E}} + \frac{\Gamma_\mathcal{L}}{\rho_\mathcal{L}}\right) dx$.

Remark 10. *Although there is no biophysical reason to impose the equality between the top velocities, this assumption remains acceptable in this context. Indeed, our focus concerns the biofilm development and the final time considered prevents the biofilm to reach the top of the domain. Therefore, in our context, the hypothesis that all top velocity are equals should not affect the dynamics of the biofilm growth.*

2.3 Synthesis of model equations

According to the previous section the PDE system under consideration writes:

$$\phi_\mathcal{A} + \phi_\mathcal{E} + \phi_\mathcal{L} = 1, \tag{6.12a}$$

$$\partial_t \phi_\mathcal{A} + \partial_x(\phi_\mathcal{A} v_\mathcal{A}) = \frac{\Gamma_\mathcal{A}}{\rho_\mathcal{A}}, \tag{6.12b}$$

$$\partial_t \phi_\mathcal{E} + \partial_x(\phi_\mathcal{E} v_\mathcal{E}) = \frac{\Gamma_\mathcal{E}}{\rho_\mathcal{E}}, \tag{6.12c}$$

$$\partial_t \phi_\mathcal{L} + \partial_x(\phi_\mathcal{L} v_\mathcal{L}) = \frac{\Gamma_\mathcal{L}}{\rho_\mathcal{L}}, \tag{6.12d}$$

$$\partial_t(\rho_\mathcal{A} \phi_\mathcal{A} v_\mathcal{A}) + \partial_x\big(\rho_\mathcal{A} \phi_\mathcal{A} v_\mathcal{A}^2\big) + \partial_x \pi_\mathcal{A} = -\phi_\mathcal{A} \partial_x P + \frac{4}{3}\partial_x\Big(\mu_\mathcal{A} \phi_\mathcal{A} \partial_x v_\mathcal{A}\Big) + F_\mathcal{A} + \Gamma_\mathcal{A} v_\mathcal{A}, \tag{6.12e}$$

$$\partial_t(\rho_\mathcal{E} \phi_\mathcal{E} v_\mathcal{E}) + \partial_x\big(\rho_\mathcal{E} \phi_\mathcal{E} v_\mathcal{E}^2\big) + \partial_x \pi_\mathcal{E} = -\phi_\mathcal{E} \partial_x P + \frac{4}{3}\partial_x\Big(\mu_\mathcal{E} \phi_\mathcal{E} \partial_x v_\mathcal{E}\Big) + F_\mathcal{E} + \Gamma_\mathcal{E} v_\mathcal{E}, \tag{6.12f}$$

$$\partial_t(\rho_{\mathcal{L}}\phi_{\mathcal{L}}v_{\mathcal{L}}) + \partial_x\big(\rho_{\mathcal{L}}\phi_{\mathcal{L}}v_{\mathcal{L}}^2\big) = -\phi_{\mathcal{L}}\partial_x P + \frac{4}{3}\partial_x\Big(\mu_{\mathcal{L}}\phi_{\mathcal{L}}\partial_x v_{\mathcal{L}}\Big) + F_{\mathcal{L}} + \Gamma_{\mathcal{L}}v_{\mathcal{L}}, \tag{6.12g}$$

where the sources terms $(\Gamma_\alpha)_\alpha$, the elastic tensors $(\pi_\alpha)_\alpha$ and the drag forces $(F_\alpha)_\alpha$ are given by:

$$\Gamma_{\mathcal{A}} = \psi_g - \psi_e - \psi_d, \qquad \Gamma_{\mathcal{E}} = \psi_e + \eta_{\mathcal{E}}\psi_d, \qquad \Gamma_{\mathcal{L}} = (1-\eta_{\mathcal{E}})\psi_d - \eta_{\mathcal{L}}\psi_g, \tag{6.13a}$$

$$\psi_g = \mu_g\rho_{\mathcal{A}}\phi_{\mathcal{A}}\phi_{\mathcal{L}}, \qquad \psi_e = \mu_e\rho_{\mathcal{A}}\phi_{\mathcal{A}}, \qquad \psi_d = \mu_d\rho_{\mathcal{A}}\phi_{\mathcal{A}}, \tag{6.13b}$$

$$\pi_\alpha = \gamma_\alpha\left(\frac{\phi_\alpha}{\phi_\alpha^\star}\right)^{\beta_\alpha}, \qquad \alpha \in \{\mathcal{A},\mathcal{E}\}, \tag{6.13c}$$

$$F_\alpha = \sum_{\alpha'\neq\alpha} f_{\alpha,\alpha'}(v_{\alpha'} - v_\alpha), \qquad f_{\alpha,\alpha'} = f_{\alpha',\alpha} \qquad \alpha \in \{\mathcal{A},\mathcal{E},\mathcal{L}\}. \tag{6.13d}$$

The system (6.12) is supplemented with the boundary conditions

$$v_\alpha(x=0) = 0, \qquad \text{and} \qquad v_\alpha(x=L) = \int_0^L \left(\frac{\Gamma_{\mathcal{A}}}{\rho_{\mathcal{A}}} + \frac{\Gamma_{\mathcal{E}}}{\rho_{\mathcal{E}}} + \frac{\Gamma_{\mathcal{L}}}{\rho_{\mathcal{L}}}\right)\mathrm{d}x,$$

for all $\alpha \in \{\mathcal{A},\mathcal{E},\mathcal{L}\}$.

The initial data for the volume faction can be chosen arbitrarily provided they are biologically relevant. However, to enforce the algebraic constraint on the sum over all the volume fractions (6.1), the velocities have to satisfy the incompressibility constraint (6.11) at all times and therefore the initial velocities must verify this constraint as well. Thus, the initial velocities are defined through a pressure P computed using the incompressibility constraint, see section 3.3, by $v_\alpha^0 = \tilde{v}_\alpha^0 - \frac{\partial_x P}{\rho_\alpha}$ where $\tilde{v}_\alpha^0$ is the initial desired velocity. Here, the system is assumed to be initially at rest so $\tilde{v}_\alpha^0 = 0$ for all the phases.

Most of the parameters come from [10] or [74, 75]. The viscosity coefficient for microalgae and the extra-cellular-matrix are taken from [73]. All the parameter values are gathered in table 6.2.

3 Numerical scheme

In this section, we are interested in the numerical approximation of the PDE system (6.12). Nevertheless, the general principles and in particular the treatment of the pseudo incompressibility constraint remain valid in a more general context. In such PDE systems, the pressure is defined through the volume filling constraint (6.1), namely $\phi_{\mathcal{A}} + \phi_{\mathcal{E}} + \phi_{\mathcal{L}} = 1$ for the considered model. The treatment of this constraint and thus the definition of the pressure is always an issue and requires specific treatment. To this end, the momentum equations are treated using a projection correction method inspired by the numerical method introduced by Chorin [18, 19, 20] and Temam [90] for incompressible viscous flows. In a nutshell, the momentum equation is decomposed using a time splitting to separate the contribution of the pressure as follows:

$$\partial_t(\rho_\alpha\phi_\alpha v_\alpha) + \partial_x\big(\rho_\alpha\phi_\alpha v_\alpha^2\big) + \partial_x\pi_\alpha = frac43\partial_x\Big(\mu_\alpha\phi_\alpha\partial_x v_\alpha\Big) + F_\alpha + \Gamma_\alpha v_\alpha, \tag{6.14a}$$

$$\partial_t(\rho_\alpha\phi_\alpha v_\alpha) + \phi_\alpha\partial_x P = 0. \tag{6.14b}$$

for $\alpha \in \{\mathcal{A},\mathcal{E},\mathcal{L}\}$.

Symbol	Name	Value	Unit
μ_g	Microalgae maximal growth rate	2	$1/\text{day}$
μ_e	Microalgae maximal ECM excretion rate	0.4	$1/\text{day}$
μ_d	Microalgae maximal death rate rate	0.2	$1/\text{day}$
μ_r	Microalgae maximal respiration rate rate	0.2	$1/\text{day}$
$\eta_{\mathcal{A}}$	Microalgae pseudo-stoichiometric coefficient	1.0	$\varnothing$
$\eta_{\mathcal{L}}$	Liquid pseudo-stoichiometric coefficient	0.96	$\varnothing$
$\eta_{\mathcal{S}}$	Substrate pseudo-stoichiometric coefficient	$8.67 \cdot 10^{-2}$	$\varnothing$
$\eta_{\mathcal{C}}$	Inorganic carbon pseudo-stoichiometric coefficient	0.146	$\varnothing$
$\eta_{\mathcal{O}}$	Oxygen pseudo-stoichiometric coefficient	0.106	$\varnothing$
$\eta_{\mathcal{E}}$	Liquid pseudo-stoichiometric coefficient for death	0.90	$\varnothing$
$K_{\mathcal{I}}$	Light parameter	0.1	$\varnothing$
τ	Light absorption coefficient for the biofilm	$2.5 \cdot 10^4$	m^{-1}
$\mathcal{I}_{\text{surf}}$	Light intensity at the surface	100	$\mu\text{mol m}^{-2}\text{s}^{-1}$
$\mathcal{I}_{\text{opt}}$	Optimal light intensity	100	$\mu\text{mol m}^{-2}\text{s}^{-1}$
$K_{\mathcal{S}}$	Substrate half saturation coefficient	$6.2 \cdot 10^{-8}$	kg/L
$K_{\mathcal{C}}$	Inorganic carbon half saturation coefficient	$4.4 \cdot 10^{-6}$	kg/L
$K_{\mathcal{O}}$	Oxygen threshold for growth	$3.2 \cdot 10^{-5}$	kg/L
$n_{\mathcal{O}}$	Oxygen exponent for growth	14	$\varnothing$
$K_{\mathcal{I}}$	Light coefficient for Haldane law	0.1	$\varnothing$
K_r	Oxygen half saturation coefficient	$1.0 \cdot 10^{-6}$	kg/L
$\theta_{\text{in},\mathcal{S}}$	Input concentration for substrate	$4 \cdot 10^{-5}$	kg/L
$\theta_{\text{in},\mathcal{C}}$	Input concentration for inorganic carbon	$10 \cdot 10^{-5}$	kg/L
$\theta_{\text{in},\mathcal{O}}$	Input concentration for oxygen	$7.2 \cdot 10^{-6}$	kg/L
$D_{\mathcal{S}}$	Diffusion coefficient for substrate	$1.47 \cdot 10^{-4}$	m^2/day
$D_{\mathcal{C}}$	Diffusion coefficient for inorganic carbon	$1.80 \cdot 10^{-4}$	m^2/day
$D_{\mathcal{O}}$	Diffusion coefficient for oxygen	$1.98 \cdot 10^{-4}$	m^2/day
$\rho_{\mathcal{A}}$	Microalgae volumetric mass density	1050	kg/m^3
$\rho_{\mathcal{A}}$	Extra-cellular matrix volumetric mass density	1050	kg/m^3
$\rho_{\mathcal{L}}$	Liquid volumetric mass density	1025	kg/m^3
$\phi^\star_{\mathcal{A}}$	Microalgae close packing threshold	0.75	$\varnothing$
$\gamma_{\mathcal{A}}$	Microalgae viscoelastic tensor coefficient	$1.2 \cdot 10^{-9}$	$\text{kg m}^{-1}\text{day}^{-1}$
$\beta_{\mathcal{A}}$	Microalgae viscoelastic tensor exponent	1	$\varnothing$
$\phi^\star_{\mathcal{E}}$	Extra-cellular matrix close packing threshold	0.75	$\varnothing$
$\gamma_{\mathcal{E}}$	Extra-cellular matrix viscoelastic tensor coefficient	$1.2 \cdot 10^{-9}$	$\text{kg m}^{-1}\text{day}^{-1}$
$\beta_{\mathcal{E}}$	Extra-cellular matrix viscoelastic tensor exponent	1	$\varnothing$
$\mu_{\mathcal{L}}$	Liquid viscosity	10^{-3}	Pa s
$\mu_{\mathcal{A}}$	Microalgae viscosity	0.25	Pa s
$\mu_{\mathcal{E}}$	Extra-cellular matrix viscosity	0.75	Pa s
$f_{\mathcal{A},\mathcal{E}}$	Friction coefficient between $\mathcal{A}$ and $\mathcal{E}$	20	$\text{kg m}^{-3}\text{day}^{-1}$
$f_{\mathcal{A},\mathcal{L}}$	Friction coefficient between $\mathcal{A}$ and $\mathcal{L}$	20	$\text{kg m}^{-3}\text{day}^{-1}$
$f_{\mathcal{E},\mathcal{L}}$	Friction coefficient between $\mathcal{E}$ and $\mathcal{L}$	20	$\text{kg m}^{-3}\text{day}^{-1}$

Table 6.2: Model parameters. The parameters come from [10, 74, 73].

3.1 Projection correction method

Let us start with the presentation of the time discretization. Let $T \in \mathbb{R}^+$ be the final time and $(t_n)_{n\geqslant 0}$ a subdivision of $[0,T]$ such that $t_n = \sum_{k=0}^{n} \Delta t_k$. Consider $\alpha \in \{\mathcal{A}, \mathcal{E}, \mathcal{L}\}$ a phase and its associated volume fraction ϕ_α and velocity v_α. Then, $\phi_\alpha^n(x)$ and $v_\alpha^n(x)$ denote, respectively, their approximation at time t_n. To shorten the notations, let us drop the space variable x and denote $\delta t = \Delta t_{n+1}$. Assuming that all the quantities are known at time t_n, the approximated solution at time $t_{n+1} = t_n + \delta t$ is computed using the following steps:

1. Update the volume fractions according to the mass balance equations (6.12b)-(6.12d):

$$\phi_\alpha^{n+1} = \phi_\alpha^n - \frac{\delta t}{\rho_\alpha}\partial_x(\phi_\alpha^n v_\alpha^n) + \frac{\delta t}{\rho_\alpha}\Gamma_\alpha.$$

2. Update the momentum equations without the contribution of the pressure term by solving the following system:

$$\begin{aligned}&\phi_\alpha^{n+1} v_\alpha^{n+\frac{1}{2}} - \phi_\alpha^n v_\alpha^n \\ &\qquad = \frac{\delta t}{\rho_\alpha}\left(-\partial_x\left(\phi_\alpha^n (v_\alpha^n)^2\right) - \partial_x \pi_\alpha^n + \frac{4}{3}\partial_x\left(\mu_\alpha \phi_\alpha^{n+1}\partial_x v_\alpha^{n+\frac{1}{2}}\right) + F_\alpha\left(\phi^{n+1}, v^{n+\frac{1}{2}}\right) + \Gamma_\alpha^n v_\alpha^n\right).\end{aligned} \tag{6.15}$$

3. Compute the pressure using the incompressibility constraint (6.3). This step is detailed in subsection 3.3.

4. Update the velocity using the pressure with:

$$v_\alpha^{n+1} = v_\alpha^{n+\frac{1}{2}} - \frac{\delta t}{\rho_\alpha}\partial_x P^{n+1}.$$

3.2 1D space discretization

Following [10], the space is discretized using staggered grids. This enables avoidance of any odd/even decoupling in the stencil of the discrete version of the system. Moreover, the use of staggered grids also allows to have or deduce naturally the quantity of interest (e. g. deduce the pressure gradient on the velocity mesh grid). Let $(x_i)_{i\in[\![0,I]\!]}$ be a regular subdivision of the domain Ω such that $x_i = i\Delta x$ with $\Delta x = \frac{L}{I}$ the mesh step. Let also define the mesh cell centers: $x_{i+\frac{1}{2}} = \left(i+\frac{1}{2}\right)\Delta x$ for $i \in [\![0, I-1]\!]$. The model variables are located:

- at the mesh cell centers for the volume fraction and the pressure: $\phi_{\alpha,i+\frac{1}{2}}$, $P_{i+\frac{1}{2}}$ for $0 \leqslant i \leqslant I-1$
- at the mesh cell edges for the velocities: $v_{\alpha,i}$ for $0 \leqslant i \leqslant I$.

Figure 6.1 gives an example of the staggered grids with the localization of model variables.

Model unknowns are discretized using a finite volume scheme. The transport terms in the mass balance equations (6.12b)-(6.12d) are written:

$$\phi_{\alpha,i+\frac{1}{2}}^{n+1} = \phi_{\alpha,i+\frac{1}{2}}^{n} - \frac{\delta t}{\rho_\alpha \Delta x}\left(\mathcal{F}_{i+1}(\phi_\alpha^n, v_\alpha^n) - \mathcal{F}_i(\phi_\alpha^n, v_\alpha^n)\right) + \frac{\delta t}{\rho_\alpha}\Gamma_{\alpha,i+\frac{1}{2}} \tag{6.16}$$

where $\mathcal{F}_i$ represents the numerical mass flux at the interface x_i, which is a function of the neighboring cells. There are multiple relevant choices for the definition of the numerical flux. For the sake of simplicity, to ensure

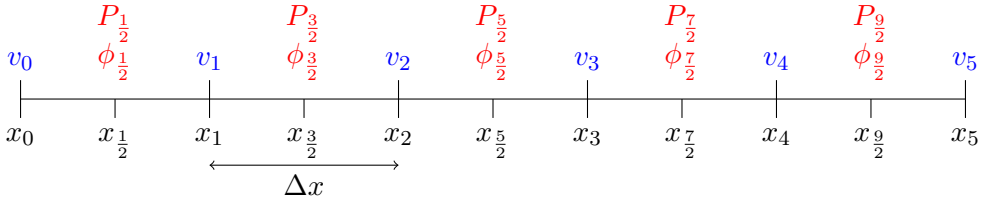


Figure 6.1: Regular staggered grid in one dimension for 5 mesh cells with the volume fractions and the velocities locations. The pressure P and the phase volume fractions $(\phi_\alpha)_\alpha$ are located at the mesh cell centers $\left(x_{i+\frac{1}{2}}\right)_{0\leqslant i\leqslant 4}$.

stability and since it is well adapted to staggered grids, it is convenient to use upwind numerical flux. Thus, the discrete mass flux is defined by $\mathcal{F}_i(\phi, v) = \mathcal{F}^+\left(\phi_{i-\frac{1}{2}}, v_i\right) + \mathcal{F}_i^-\left(\phi_{i+\frac{1}{2}}, v_i\right)$ with

$$\mathcal{F}^+(\phi, v) = \begin{cases} 0 & \text{if } v \leqslant 0, \\ \phi v & \text{if } v > 0, \end{cases} \qquad \text{and} \qquad \mathcal{F}^-(\phi, v) = \begin{cases} \phi v & \text{if } v < 0, \\ 0 & \text{if } v \geqslant 0. \end{cases}$$

All the volume fractions are updated using equation (6.16). Thus, the volume-filling constraint enforcement is not guaranteed and depends on the strategy used to compute the pressure, see sections 3.3 and 4.2.

Remark 11. *To update the volume fractions and ensure volume-filling constraint enforcement another strategy consists to use equation* (6.16) *for all the components except one (usually the liquid) which is computed using the algebraic volume-filling constraint* (6.1)*: $\phi_{\alpha'} = 1 - \sum_{\alpha\neq\alpha'} \phi_\alpha$ as done in [21, 22, 74, 75].*

For the momentum balance equation, following [10], the transport term is also discretized using an upwind strategy based on the material velocity v, that is the momentum flux is defined by

$$\mathcal{G}_{i+\frac{1}{2}} = \frac{v^n_{\alpha,i}}{2}\left(\mathcal{F}^+\left(\phi_{i-\frac{1}{2}}, v_i\right) + \mathcal{F}^+\left(\phi_{i+\frac{1}{2}}, v_{i+1}\right)\right) + \frac{v^n_{\alpha,i+1}}{2}\left(\mathcal{F}^-\left(\phi_{i+\frac{1}{2}}, v_i\right) + \mathcal{F}^-\left(\phi_{i+\frac{3}{2}}, v_{i+1}\right)\right).$$

The other terms of equation (6.15) are discretized using standard approximations. Remark that interpolation on the dual mesh is required only for the zeroth order terms like the momentum supply induced by mass exchanges or friction forces. For these terms, the approximation of the volume fraction on the dual mesh is obtained by approximating the volume fractions using the values in the neighboring cells: $\phi_i = \frac{1}{2}\left(\phi_{i-\frac{1}{2}} + \phi_{i+\frac{1}{2}}\right)$. Therefore, dropping the α for readability, equation (6.15) is discretized as follows:

$$\begin{aligned} \phi_i^{n+1} v_i^{n+\frac{1}{2}} - \frac{4\delta t}{3\Delta x^2}\mu\left(\phi_{i+\frac{1}{2}}^{n+1} v_{i+1}^{n+\frac{1}{2}} - \left(\phi_{i+\frac{1}{2}}^{n+1} + \phi_{i-\frac{1}{2}}^{n+1}\right) v_i^{n+\frac{1}{2}} + \phi_{i-\frac{1}{2}}^{n+1} v_{i-1}^{n+\frac{1}{2}}\right) - \delta t F\left(\phi_i^{n+1}, v_i^{n+\frac{1}{2}}\right) \\ = \phi_i^n v_i^n - \frac{\delta t}{\Delta x \rho}\left(\mathcal{G}_{i+\frac{1}{2}} + \pi\left(\phi^n_{i+\frac{1}{2}}\right) - \mathcal{G}_{i-\frac{1}{2}} - \pi\left(\phi^n_{i-\frac{1}{2}}\right)\right) + \delta t \Gamma(\phi_i^n) v_i^n. \end{aligned}$$

Remark 12. *In this projection step, the viscosity and the friction are treated implicitly. For the viscosity, this treatment enables the relaxation of the CFL constraint and avoids numerical instabilities.*

Remark 13. *Like in [22, 74, 75] the computation of friction forces requires a specific treatment. Indeed, the friction forces depend on the difference between the phase velocities, and when a phase vanishes the velocity can not be deduced from the momentum (ie. ϕv). In the considered applications, areas of pure liquid or biofilm are important so the adaptation of the initial data to avoid phase vanishing is irrelevant. To overcome this*

difficulty, a strategy consists to treat these terms implicitly so the velocity can be directly computed using the above equation. However, this is costly because it imposes to solve at each time step a linear system of size: number of phases×mesh grid size.

Finally, the space discretization of the correction step is given by: $v_\alpha^{n+1} = v_\alpha^{n+\frac{1}{2}} - \frac{\delta t}{\rho_\alpha \Delta x}\left(P_{i+\frac{1}{2}}^{n+1} - P_{i-\frac{1}{2}}^{n+1}\right)$

3.3 Pressure approximation

Let us detail the third step of the projection correction method. This is the key step to enforce the algebraic constraint on the sum over all the volume fractions (6.1). The standard strategy consists in plugging the time discrete version of equation (6.14b): $\phi_\alpha^{n+1} v_\alpha^{n+1} = \phi_\alpha^{n+1} v_\alpha^{n+\frac{1}{2}} - \frac{\delta t}{\rho_\alpha}\phi_\alpha^{n+1}\partial_x P^{n+1}$ into the incompressibility constraint (6.11) to obtain the following equation on the pressure:

$$\partial_x\left(\sum_\alpha \phi_\alpha^{n+1} v_\alpha^{n+\frac{1}{2}} - \frac{\delta t}{\rho_\alpha}\phi_\alpha^{n+1}\partial_x P^{n+1}\right) = \sum_\alpha \frac{\Gamma_\alpha}{\rho_\alpha} \tag{6.17}$$

Thus the pressure can be obtained by solving a non-linear and inhomogeneous Poisson equation. As mentioned above, this strategy relies on the use of the continuous version of the incompressibility constraint. Therefore, there is no guarantee that the algebraic volume-filling constraint will be fulfilled at the discrete level.

To enforce the algebraic volume filling constraint, we adapt the strategy proposed in [10], which consists in using the fully discretized mass balance equations to deduce the appropriate discrete incompressibility constraint. To this end, let us assume that the constraint $\sum_\alpha \phi_{\alpha,i+\frac{1}{2}}^n = 1$ is satisfied for all times $(t_n)_{n\geqslant 0}$ and in all the grid mesh cells. Thus, the sum of the equations (6.16) over the phases leads to

$$\sum_\alpha \frac{1}{\rho_\alpha}\left(\mathcal{F}_{i+1}(\phi_\alpha^n, v_\alpha^n) - \mathcal{F}_i(\phi_\alpha^n, v_\alpha^n)\right) = \Delta x \sum_\alpha \frac{\Gamma_{\alpha,i+\frac{1}{2}}}{\rho_\alpha}. \tag{6.18}$$

Then, as in the standard strategy, an equation on the pressure or its gradient can be deduced by using the time discrete version of equation (6.14b). Since in the correction step, the volume fractions remain unchanged, the time discrete version of equation (6.14b) simplifies into $v_\alpha^{n+1} = v_\alpha^{n+\frac{1}{2}} - \frac{\delta t}{\rho_\alpha}\partial_x P^{n+1}$. Injecting this relation into equation (6.18) gives:

$$\sum_\alpha \frac{1}{\rho_\alpha}\left(\mathcal{F}_{i+1}\left(\phi_\alpha^n, v_\alpha^{n+\frac{1}{2}} - \frac{\delta t}{\rho_\alpha}\partial_x P^{n+1}\right) - \mathcal{F}_i\left(\phi_\alpha^n, v_\alpha^{n+\frac{1}{2}} - \frac{\delta t}{\rho_\alpha}\partial_x P^{n+1}\right)\right) = \Delta x \sum_\alpha \frac{\Gamma_{\alpha,i+\frac{1}{2}}}{\rho_\alpha}. \tag{6.19}$$

Consequently, to ensure that the algebraic volume-filling constraint is met, the pressure must be the solution of the non-linear equation (6.19). The solution can be approximated using Newton's methods. In practice, although this method is more expensive than the standard approach its cost remains reasonable. Indeed, the Jacobian matrix is explicitly known and the solution at the previous time step reveals to be a good initial guess so only very few iterations are necessary to converge. Both strategies are compared in subsection 4.2.

4 Numerical results

The aim of the paper is to present and test a numerical method able to simulate mixture models for biofilms by guaranteeing the preservation of the algebraic volume filling constraint. Another challenge when one wants to go towards the applications, relies on the difficulty to calibrate the parameters of the model. Many parameters are, up to our knowledge, not available in the current literature and very difficult to extrapolate from experimental data. For example, in [21, 22, 74, 75] the elastic tensor (ie. π_α) settings are calibrated so that the biofilm front velocity matches observations. Consequently, any modification of the model requires recalibration. To avoid such difficulties, subsection 4.1 presents numerical simulations based on the numerical scheme presented in section 3, but assumes that the viscosity can be neglected, which enables reusing parameters from [21, 22, 74, 75] for the elastic tensors. Secondly, subsection 4.2 presents comparisons between the two strategies to approximate the pressure, still neglecting the viscosity. Finally, subsection 5.1 presents the dynamic of the full model including viscosity and recalibration of the elastic tensors.

Initially, the mixture is only made of microalgae and liquid and the volume fractions are set by

$$\phi_{\mathcal{A}}^0(x) = \max\{0, 0.05(x-0.1)(x+0.1)\}, \quad \phi_{\mathcal{E}}^0 = 0, \quad \text{and} \quad \phi_{\mathcal{L}}^0 = 1 - \phi_{\mathcal{A}}. \tag{6.20}$$

As mentioned in subsection 2.3, the system is assumed to be at rest. Thus, the initial velocities are defined by $v_\alpha^0 = -\frac{1}{\rho_\alpha}\partial_x P$ where the pressure P is determined according to the strategy presented in subsection 3.3 to enforce the algebraic constraint on the sum over all the volume fractions (6.1).

4.1 Biofilm dynamic without viscosity

Figure 6.2 presents the numerical results for different times of system (6.12) where the viscous terms are neglected, namely $\mu_\alpha = 0$ for $\alpha \in \{\mathcal{A}, \mathcal{E}, \mathcal{L}\}$. The simulation is made using the numerical scheme presented in section 3 and using the strategy based on the adaptation of [10] for the computation of the pressure, see subsection 3.3. In these figures, the left side corresponds to the surface where the biofilm sticks and develops and the right side corresponds to the side covered by the liquid, where nutrients are brought.

According to Figures 6.2a, 6.2b and 6.2c, there is front propagation corresponding to the biofilm (dashed orange curve) development within the liquid. As in [74], two areas can be distinguished within the biofilm. For example in Figure 6.2c, on the left side, namely for $x \in [0, \sim 6]\mu\text{m}$, the biofilm is mainly made of extra-cellular matrix (ie. $\mathcal{E}$), whereas on the biofilm front, namely for $x \in [\sim 6, 7.6]\mu\text{m}$, the biofilm is mainly made of microalgae. On the opposite, the right side, namely for $x > 7.6$, is made of pure water.

For the biofilm components, the velocities are positive near the front, which is expected and explains the biofilm expansion. Otherwise, the liquid velocity is negative in the biofilm region, which means that the liquid is drained into the biofilm due to its consumption for the biofilm growth.

4.2 Volume filling constraint validation

Let us compare the two strategies presented in subsection 3.3 to enforce the algebraic volume-filling constraint, that is computing the pressure P either as the solution of discretization the linear equation (6.17) (standard strategy), or as the solution of the non-linear equation (6.19) (adapted strategy). To this aim, as in subsection 4.1, the system (6.12) without the viscous terms is simulated, but using the standard strategy to enforce the volume

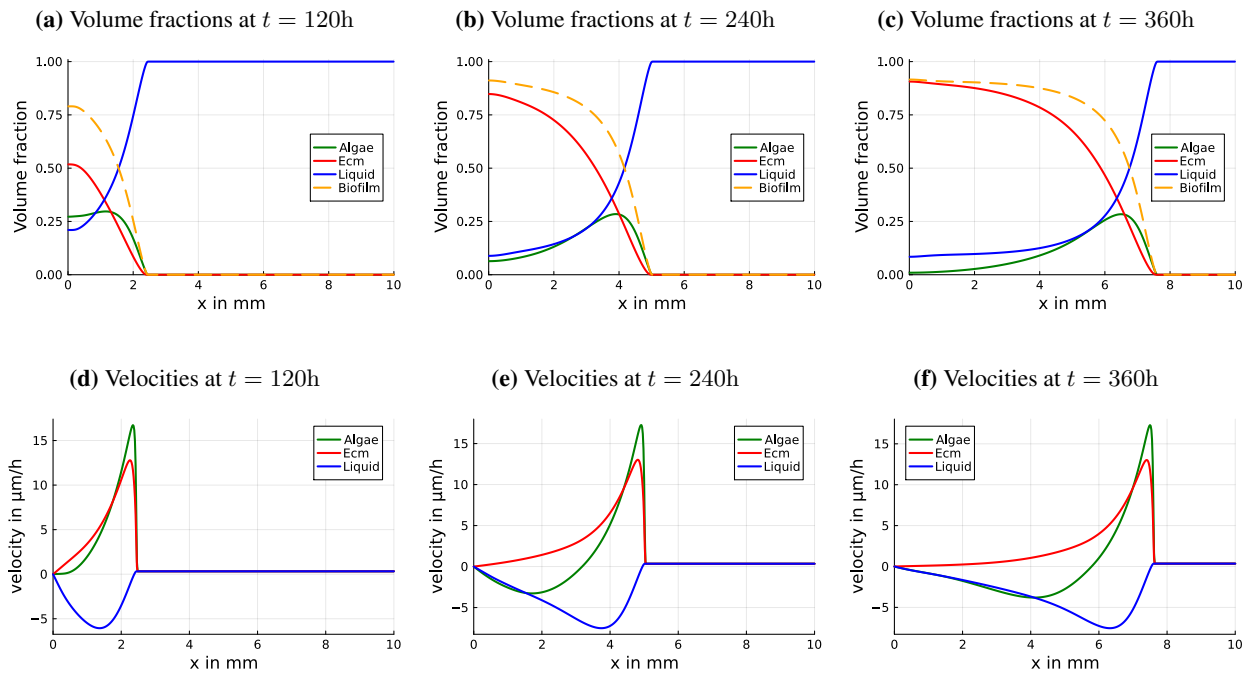


Figure 6.2: Mixture components volume fractions (first row) and velocities (second row) for different times.

filling constraint. The results for the mixture components volume fractions at $t = 360h$ are presented in Figure 6.3a and can be compared to Figure 6.2c. According to these plots, the results are comparable. Similarly, the shape of the pressure gradients curves are also similar, see Figure 6.3b. Nevertheless, according to the dotted purple curve in Figure 6.3b, there is a significant discrepancy close to the biofilm front (ie. at $x = 7.6mm$) in the pressure gradients between the two correction strategies. Note that the front is the active part of the biofilm. Namely, it is in this area that the source terms are the largest and lead to significant changes in mixture composition. Therefore, it is expected that the effect of the pressure gradient is observable notably there.

Besides, the pressure can be interpreted as the Lagrange multiplier associated with the volume-filling constraint. Thus, it is important to compare how these strategies enable enforcing at the discrete level the volume filling constraint (6.1). To this end, Figure 6.3c represents the sum of the volume fractions within the domain a time $t = 360h$. According to this plot, the strategy adapted from [10] enables ensuring the volume filling constraint, whereas the standard strategy does not. Numerically, the maximal error on the volume-filling constraint for the standard strategy is $1.007 \cdot 10^{-3}$, whereas with the adapted strategy, it is $5.107 \cdot 10^{-15}$, namely the order of magnitude of the precision used in Newton's method. Moreover, with this adapted method, the error remains negligible throughout the simulation whereas, with the standard strategy, it varies over time, see Figure 6.7 in Appendix 6.

5 Model extensions

Following insights coming from [74], this section presents various relevant extensions of the model and their numerical simulations.

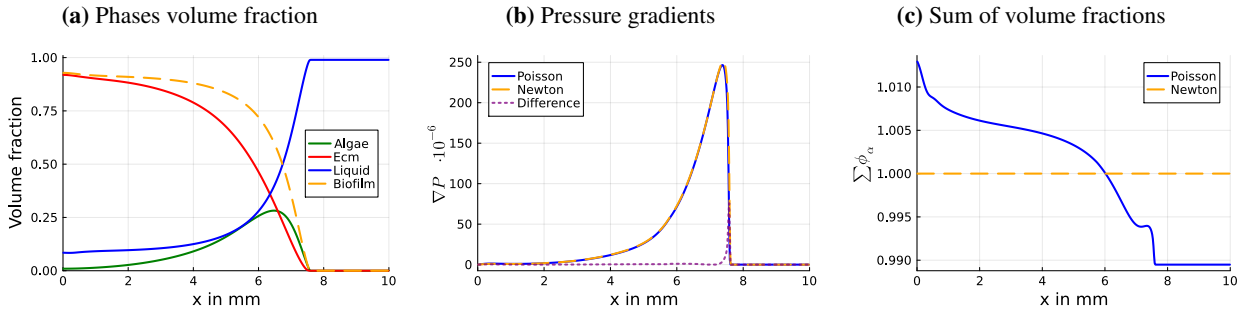


Figure 6.3: Mixture components volume fractions (left), pressure gradients (center) and the sum of the volume fractions (right) at time $t = 360h$. In figures 6.3b and 6.3c the blue curve represents correspond to a simulation made using the standard strategy for the pressure gradient computation (ie. solving a Poisson equation) and the dashed orange curve represents the results obtained using the strategy adapted from [10] which require the resolution of a non-linear equation (ie. Newthon's method).

5.1 Including the viscosity

Adding the viscous terms for the components requires recalibrating the model parameters. Indeed, the viscosity is a measure of the component's resistance to deformation. Therefore, when accounting for the viscosity, the parameters associated with the component's ability to deform must be adapted. In particular, the elastic tensors for the microalgae and the extra-cellular matrix must be recalibrated. Moreover, up to our knowledge, there is no direct measurement of the parameters and they are calibrated, see [21, 22, 74], such that the biofilm front velocity matches experimental measurements, see [87]. However, such calibration is extremely complex because the biofilm front velocity depends also on many other parameters like the growth or death rate. Nevertheless, to get the right order of magnitude of the biofilm front velocity the elastic tensor coefficients must be significantly increased: multiplied by $9 \cdot 10^7$ so is set to $\gamma_{\mathcal{A}} = \gamma_{\mathcal{E}} = 4.5 \cdot 10^{-3}\mathrm{kg/m/day}$.

Figure 6.4 represents the time dynamic of mixture components when accounting for the contribution of viscosity. The global dynamic remains comparable to the dynamic observed in Figure 6.2. In particular, there is still a biofilm traveling front. Again, there are two areas within the biofilm: the back which is mainly made of an extra-cellular matrix (for $x \in [0, 5]$ in subfigure 6.4c), and the front which is mainly made of microalgae (for $x \in [3.97, 6.7]\mu\mathrm{m}$ in subfigure 6.6c). Nevertheless, a major discrepancy is that the microalgae remain more located at the front when including the viscosity. This is particularly visible at $t = 240\mathrm{h}$ when comparing Figure 6.2b and Figure 6.4b. In addition, at $t = 240\mathrm{h}$, the velocities order of magnitude close to the front is larger when including the viscosity. However, the interpretation of this observation is tricky. Indeed, the shift in the elastic tensors for the biological phases imposes the use of very refined mesh grids to properly capture the biofilm dynamic. Thus, it would be of particular interest to design and use well-balanced numerical scheme able to preserve the biofilm front structure. For more details about the numerical convergence of the scheme, see appendix 6.

5.2 Including light intensity

A microalga is a photosynthetic organism. Thus, microalgae require light to grow. When microalgae develop within a biofilm, the upper layers overshadow the lower layers. Following [21, 22, 74] to account for these mechanisms, the microalgae growth rate becomes $\psi_g = \mu_g \rho_{\mathcal{A}} \phi_{\mathcal{A}} \phi_{\mathcal{L}} f_I$, where f_I accounts for the effect of light

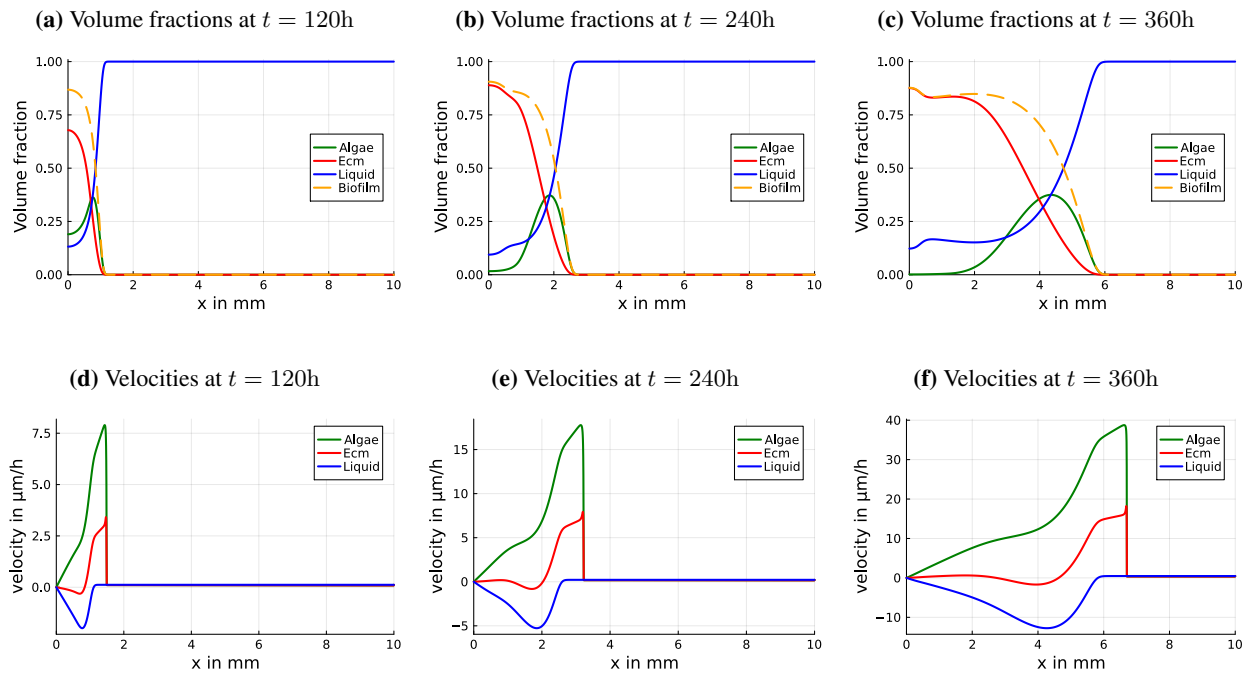


Figure 6.4: Mixture components volume fractions (first row) and velocities (second row) for different times. In this simulation, the viscosity is included and the elastic tensors for the biological phases are multiplied $9 \cdot 10^7$ so the biofilm front velocity matches experimental measurements. The simulation is made using 2048 mesh cells for the space grid.

on growth. This term depends on the rescaled received light intensity I and takes the form of the Haldane law:

$$f_I = \frac{2(1 + K_{\mathcal{I}})\mathcal{I}}{\mathcal{I}^2 + 2K_{\mathcal{I}}\mathcal{I} + 1}. \tag{6.21}$$

The rescaled light intensity is the ratio between the received light and the optimal light intensity $\mathcal{I}_{\text{opt}}$, namely:

$$\mathcal{I}(t, x) = \frac{\mathcal{I}_{\text{surf}}}{\mathcal{I}_{\text{opt}}} \exp\left(- \int_x^L \tau\big(1 - \phi_{\mathcal{L}}(t, y)\big) dy \right), \tag{6.22}$$

where I_{surf} is the light intensity at the surface of the tank (ie. $x = L$) and τ the attenuation coefficient of the biofilm, assuming that microalgae and extra-cellular matrix have the same attenuation rate. The parameter values associated to the light are gathered in Table 6.2.

Figure 6.5 represents the time dynamic of mixture components when accounting for the contribution of light. The global dynamic is comparable to the dynamic observed in Figure 6.2. In particular, the biofilm front position travels at the same speed, and, again, there are two areas within the biofilm: the back which is mainly made of an extra-cellular matrix (for $x \in [0, 6]$ in subfigure 6.5c) and the front which is mainly made of microalgae (for $x \in [6, 7.6]\mu\text{m}$ in subfigure 6.5c). However, as expected, the volume fraction of biofilm is lower. Indeed, taking into account the effect of light reduces the growth in the shadowed areas and thus the biomass of microalgae. The extra-cellular matrix is also reduced since it is made from microalgae excretion and dead organisms.

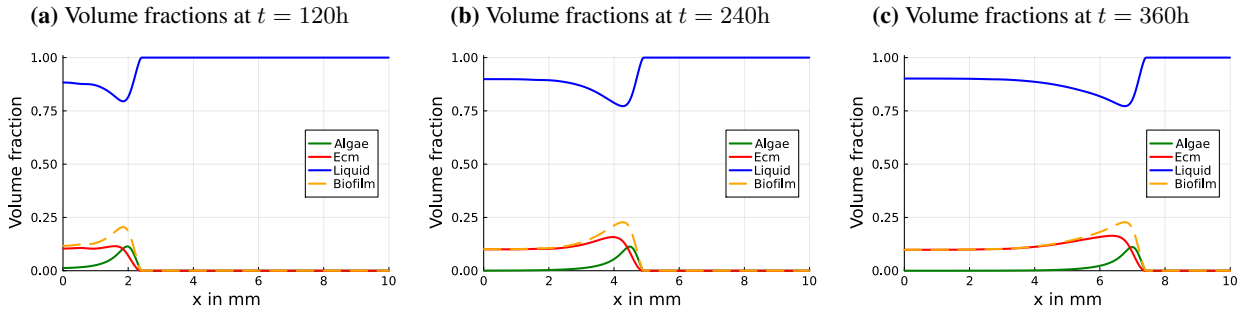


Figure 6.5: Mixture components volume fractions for different times. In this simulation, the effective microalgae growth rate (ie. ψ_g) accounts for the contribution of light intensity through Haldane's law (6.21) and light attenuation induced by biofilm layers, see equation (6.22)

5.3 Including light intensity and solutes

Following [74], let us now include three dissolved components: the substrate ($\mathcal{S}$), the inorganic carbon ($\mathcal{C}$), and the oxygen ($\mathcal{O}$). As mentioned in subsection 2.1, the dynamic for dissolved components is modeled using a convection-diffusion reaction equation (6.9).

In a nutshell, the substrate represents the nitrate which is a nutrient of primary importance for the growth of autotrophic organisms like microalgae. Besides, roughly speaking, photosynthesis is the assimilation of inorganic carbon using light energy by autotrophic organisms. Photosynthesis releases oxygen. Thus, including these components is of primary interest. Taking into account these compounds also allows us to include the process of respiration. Basically, respiration is the opposite mechanism of photosynthesis and its consideration allows us to better describe the dynamic of thick biofilms. Indeed, the process of respiration becomes non-negligible in the absence of light, namely in the biofilm's inner layers.

As for the light, the contributions of the dissolved components to the photosynthesis process are accounted for in the growth through the multiplication by functions f_p, $p \in \{\mathcal{S}, \mathcal{C}, \mathcal{O}\}$ which represent how the growth is modified by the local concentration of these components. On the one hand, limited contribution in high concentration regimes of the substrate and the inorganic carbon is modeled using Monod's law: $f_p = \frac{\theta_p}{K_p+\theta_p}$. On the other hand, the inhibition induced by high oxygen concentration is modeled by the sigmoidal function $f_{\mathcal{O}} = \frac{1}{1+\left(\frac{\theta_{\mathcal{O}}}{K_{\mathcal{O}}}\right)^{n_{\mathcal{O}}}}$. Thus, including the contribution of the dissolved components and the light intensity, the algae growth rate becomes: $\psi_g = \mu_g \rho_{\mathcal{A}} \phi_{\mathcal{A}} \phi_{\mathcal{L}} f_I f_{\mathcal{S}} f_{\mathcal{C}} f_{\mathcal{O}}$.

The respiration process is modeled by $\psi_r = \mu_r \phi_{\mathcal{A}} \frac{\theta_{\mathcal{O}}}{K_r+\theta_{\mathcal{O}}}$ where μ_r is the maximal respiration rate and K_r the half-saturation constant for the oxygen.

The modification of the microalgae growth rate and the inclusion of the respiration process requires to adapt the source terms for the phases as follows:

$$\Gamma_{\mathcal{A}} = \psi_g - \psi_e - \psi_d - \psi_r, \qquad \Gamma_{\mathcal{E}} = \psi_e + \eta_{\mathcal{E}} \psi_d, \qquad \Gamma_{\mathcal{L}} = (1 - \eta_{\mathcal{E}})\psi_d + \eta_{\mathcal{L}}(\psi_r - \psi_g).$$

As for a phase, the source terms for a dissolved component is the sum of the pseudo-stoichiometric coefficients multiplied by the reaction rates. Thus, for the dissolved components, the source terms are

$$\Gamma_{\mathcal{S}} = -\eta_{\mathcal{S}} \psi_g, \qquad \Gamma_{\mathcal{C}} = -\eta_g^{\mathcal{C}} \psi_g + \eta_r^{\mathcal{C}} \psi_r, \qquad \Gamma_{\mathcal{O}} = \eta_g^{\mathcal{O}} \psi_g - \eta_r^{\mathcal{O}} \psi_r.$$

The external supply for the dissolved components is modeled through Dirichlet boundary conditions at the top of the bioreactor, namely at $x = L$. Otherwise, the no flux boundary condition at the bottom of the bioreactor is modeled using the Neumann boundary condition: $\partial_x \theta_p|_{x=0} = 0$ for $p \in \{\mathcal{S}, \mathcal{C}, \mathcal{O}\}$.

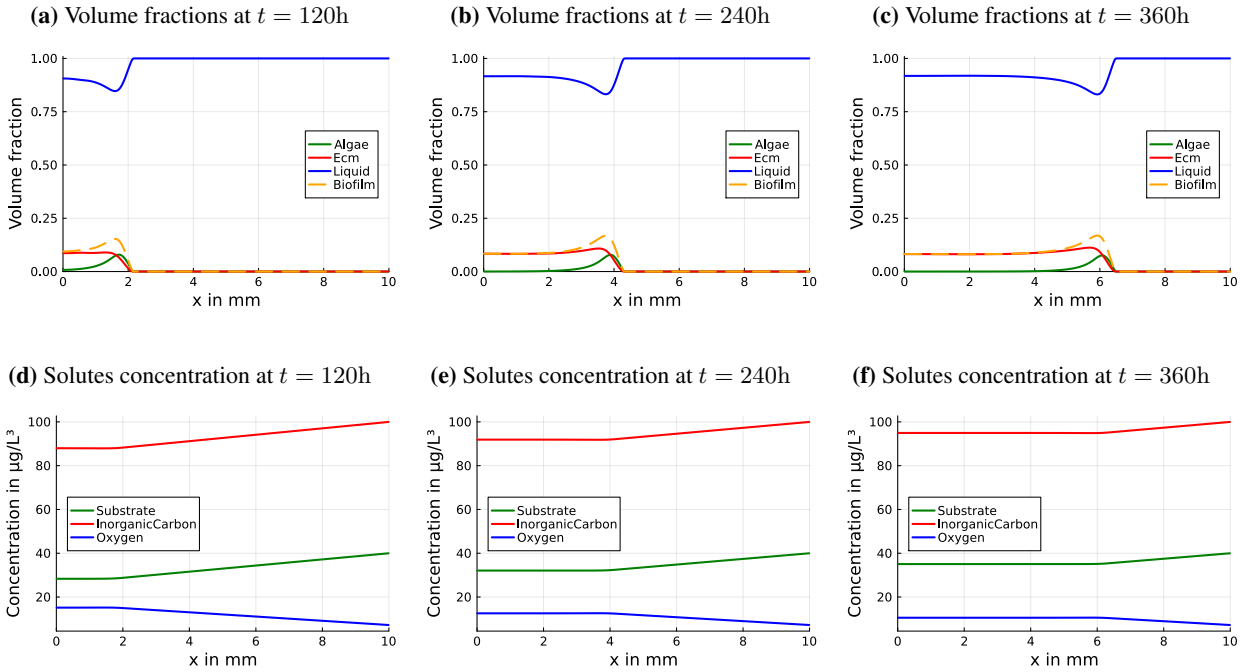


Figure 6.6: Mixture components volume fractions (first row) and velocities (second row) for different times.

The parameter values associated with the inclusion of the dissolved components are gathered in Table 6.2.

Numerically, the transport and reaction terms in the mass balance equations for the solutes are treated similarly to the other components. The diffusion terms are treated implicitly to ensure stability without constraining the CFL condition.

Figure 6.6 represents the time dynamic of mixture components when accounting for the contribution of light and solutes. The global dynamic is comparable to the dynamic observed in Figure 6.5. In particular, the biofilm front position travels at a comparable speed. Again there are two areas within the biofilm: the back which is mainly made of an extra-cellular matrix (for $x \in [0, 6.5]$ in subfigure 6.6c), and the front which is mainly made of microalgae (for $x \in [5.5, 6.6]\mu$m in subfigure 6.6c). However, the biofilm front velocity is slightly slower here. This can be explained by the fact that the lack or excess of solutes in the active part of the biofilm slightly reduces its growth. Indeed, for example, at $t = 360h$, within the biofilm area, the concentration of substrate is reduced by 12.4% and the concentration of inorganic carbon is reduced by 5.1% relatively to the input values (ie. θ_{in}). Besides, the concentration of oxygen is increased by 46.3% relatively to $\theta_{in,\mathcal{C}}$. These discrepancies are larger at the beginning and tend to decrease over time, see Figure 6.9 in the supplementary material. These results are in good agreement with the results presented in [74, 75].

6 Conclusions and perspectives

This article proposes an adaptation of the numerical scheme presented in [10] able to enforce the volume filling constraint in mixture models including mass exchanges. As in [10] the strategy consists in deducing the discrete version of the incompressibility constraint from the discretized mass balance equations. Numerical simulations show that this method enables the enforcement of the total volume filling constraint at the discrete level.

In addition, on the modeling side, previous models from the literature are enriched by the inclusion of viscous terms. These terms are essential to properly model biofilms in their fluidic environment especially when there is a mixing of the surrounding fluid. In this context, this work has allowed us to highlight the importance of designing well-balanced numerical scheme able to efficiently capture the biofilm dynamic when including the viscosity. Indeed, including the viscosity requires to recalibrate model parameters; in particular the elastic tensors need to be strongly rescaled in order to recover realistic front features. However, with these parameters, the numerical set up is more demanding to reach convergence. This difficulty leads to consider further the design of a specific well-balanced scheme for the problem. To this end, the use of well-balanced numerical schemes able to preserve the equilibrium at the biofilm front can be considered.

Finally, in further works, it would be interesting to include additional biological features. Among others, biofilms are generally multi-species. The framework of mixture theory is well adapted to incorporate different species and such extensions are affordable if the interaction between the species and their metabolisms is known. To make the model even more realistic and predictive its calibration on experimental data is also particularly interesting. In conclusion, real-life biofilms are 3D and therefore the extension and implementation of the numerical method in 2D and 3D should be considered.

Acknowledgement

- This work was supported by Conseil Regional de Bourgogne Franche-Comté (France).

Appendix

Spatiotemporal equilibrium

The spatiotemporal equilibrium states for the system (6.12) correspond to the state solution where the source terms of all phases vanish, namely: $\Gamma_i = 0$ for $i \in \{\mathcal{A}, \mathcal{E}, \mathcal{L}\}$. In particular, $\Gamma_\mathcal{E} = 0$ induces $\psi_e + \eta_\mathcal{E}\psi_d = 0$ which lead to $\phi_\mathcal{A} = 0$. Thus, the only spatiotemporal stationary state is the null state, namely $\phi_i = 0$ for $i \in \{\mathcal{A}, \mathcal{E}, \mathcal{L}\}$.

Time dynamic of the volume fraction sum

Numerical convergence analysis for the model including viscosity

Numerical experiments have shown that numerical parameters need to be significantly reduced to reach convergence when the viscosity is included, and re-estimating the elastic tensor accordingly. Indeed, as mentioned in section 5.1, when including the viscosity, the elastic tensor coefficients must be rescaled and multiplied by $9 \cdot 10^7$

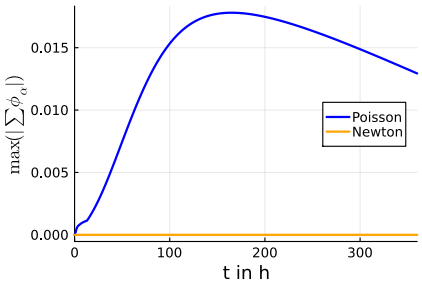


Figure 6.7: Time evolution for the maximal error within the domain on the sum of volume fractions: $E = \max_x |\sum_\alpha \phi_\alpha - 1|$.

to obtain realistic front velocities for the biofilm. Figure 6.8 shows the convergences of the numerical scheme in both cases: with and without the viscous term. As expected, numerical convergence is obtained in both cases. Nevertheless, as presented in Figure 6.8 the convergences rate is lower when the viscous term is included. This explains at least partially why the numerical parameters need to be significantly reduced to reach acceptable precision for the application considered when the viscosity is included

(a) Numerical convergence for the model without the viscous term.

(b) Numerical convergence for the model incluging the viscous term.

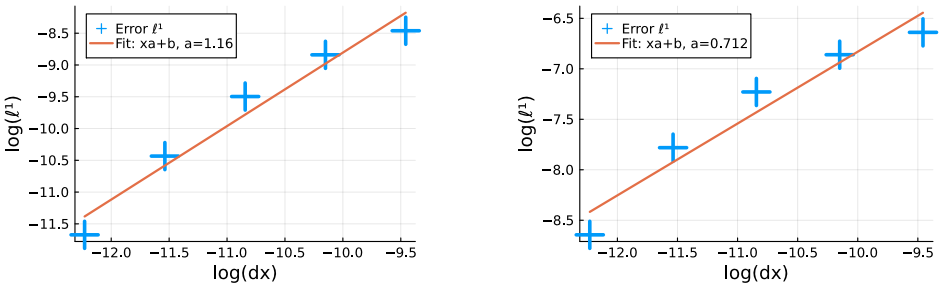


Figure 6.8: Numerical analysis of the convergence of the scheme. The left side correspond to the case without the viscous term, namely $\mu_\alpha = 0$ for $\alpha \in \{\mathcal{A}, \mathcal{E}, \mathcal{L}\}$ and the right side correspond to the case with the viscous term and using very large values for the elastic tensor coefficients.

Relative variation of solutes concentration

Figure 6.9 shows the relative variation of solutes concentration for different times associated to the simulation presented in subsection 5.3. In this figure, we observe that the variations relatively to the input concentration are larger at the beginning (ie. $t = 120$h) than at the end (ie. $t = 360$h).

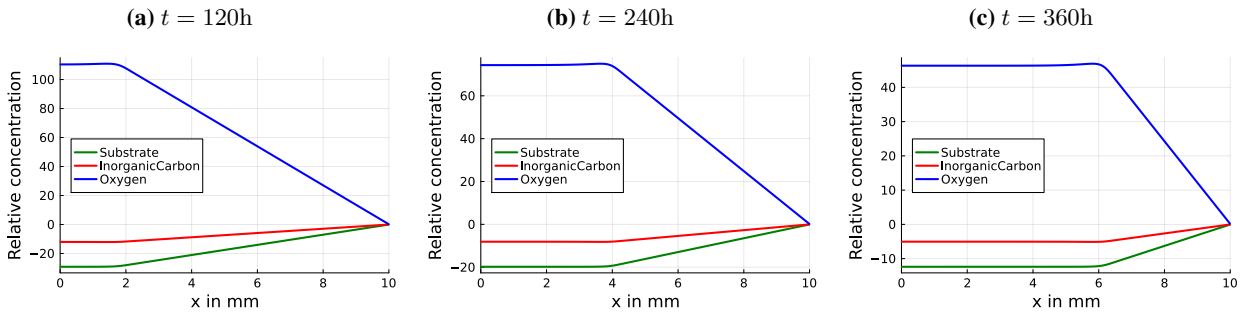


Figure 6.9: Solutes relative concentration for different times

Conclusion

What about the continuous problem?

Our control problem is formulated as a spatially discretized model, which is classically called a "discretize then optimize" approach. It would be interesting to compare it with the "optimize then discretize" approach, which consists in using a continuous optimization algorithm that is then applied to a discretized model. This would raise the question of the existence and uniqueness of optimal control, as well as its regularity, and it would also be interesting to study first-order optimality conditions in this framework. Furthermore, we have shown that for any spatial discretization step Δx, the associated optimal control problem $\mathcal{P}_{\Delta x}$ has a solution $u_{\Delta x}$. This leads to the following natural questions, related to what happens when this step shrinks to zero:

- Does $u_{\Delta x}$ converge to some u_0 when $\Delta x \to 0$?
- What is the nature of this convergence?
- Is there a topological dependence of the control on Δx?

A more general question concerns the Γ-convergence of solutions (see e.g. [4, 12]), which would imply at the same time the convergence of minimizers as $\Delta x \to 0$, and the one of the value function. This question would help justify the use of a discrete model as a good approximation to the continuous model derived from Coclite et. al. [23]. It is also expected that a better understanding of the continuous problem will not only provide the basis for new algorithms, but also guarantee the convergence and stability of solutions.

Scaling the problem up

An intriguing research objective lies in the exploration of strategies for optimizing traffic flow within sprawling metropolitan road networks. However, a formidable obstacle currently looms large, primarily related to the substantial computational resources demanded by the control algorithm discussed in chapter 2 when dealing with extensive network configurations.

Starting with the extension of the deep hyper-reduction approach to 1x2, 2x1 and 2x2 junctions, the synthesis of the main elements of this thesis into a coherent methodology could be an interesting way to tackle this numerical challenge. In this approach, we would construct a simplified model tailored to accommodate the inherent complexities of significantly larger networks, thereby mitigating the computational demands. Within the framework of this lightweight but robust model, we would conduct controlled experiments to discern effective traffic control strategies.

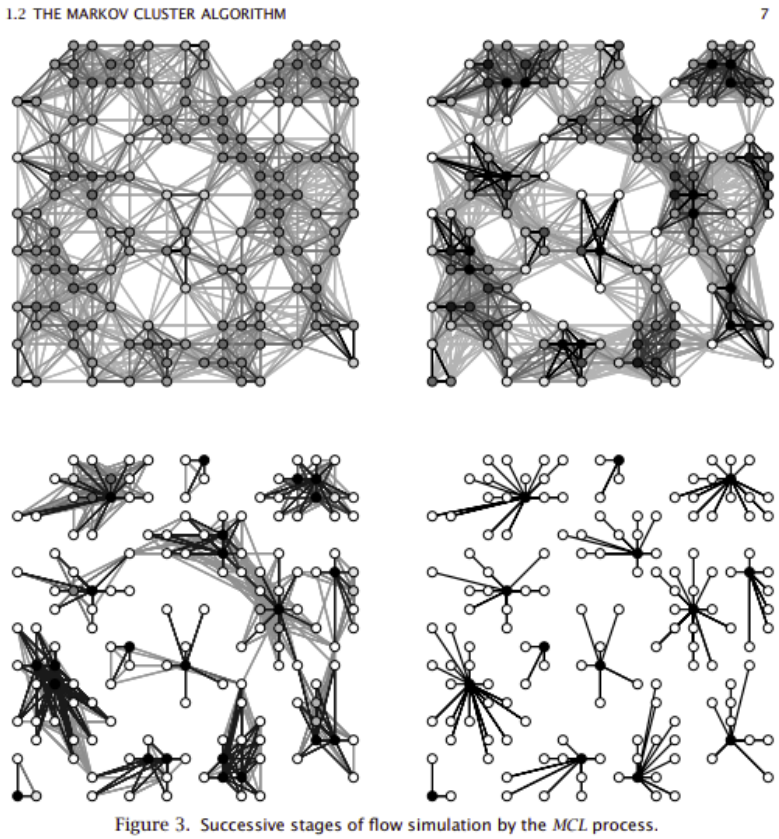


Figure 6.10: Example of graph reduction using the Markov clustering algorithm in the thesis of *S. V. Dongen (2000)*.

Afterwards, we hope that the findings and lessons learned from these controlled experiments using the scaled-down model can potentially be extended and used to address the practical issues of traffic management in metropolitan areas. This approach aims to bridge the gap between theoretical research and real-world applications, potentially leading to enhanced efficiency in managing complex urban traffic networks.

It is paramount to emphasize that the successful implementation of this methodology relies on a thorough exploration of computational methodologies, modeling techniques and validation protocols, guaranteeing both its feasibility and its effectiveness in the field of real-world metropolitan traffic optimization.

A complementary way of scaling would be to design a meta-graph of a complex network to reduce the number of paths to be considered in the control loop. We can imagine considering a precise network only in a neighborhood of the axis to be evacuated and factorizing the rest of the graph to obtain non-physical meta-routes, but computing the appropriate boundary conditions to solve the control problem. This type of partitioning was studied in particular in S. V. Dongen's thesis [32], where he introduced the *Markov Clustering Process* (MCL process). This consists of empirically averaging random flows traversing a graph and grouping the most frequently connected nodes into a single class. In fact, such classes appear to be sufficiently isolated from the rest of the graph that they can be treated as hypersummits, which would greatly reduce the size of the problem. The Figure 6.10 show us the reduction of a 150-node graph to 14 simple subgraphs, which we could imagine treating as meta-summits or independent subgraphs, at least with greatly reduced coupling to allow efficient parallelization of computations.

Beyond road traffic

As pointed out by Ciro D'Apice, Rosanna Manzo, and B. Piccoli in [30], the way we model traffic on road networks shares similarities with how we model telecommunication networks or logistical supply chains. Therefore, it could be worthwhile to explore the possibility of applying the tools developed in this thesis to these contexts. This could help in investigating issues related to network congestion and finding optimal ways to efficiently utilize the network links, all within a resource-efficient approach that minimizes the resources needed for their operation.

Bibliography

[1] S. E. Ahmed, S. Pawar, O. San, A. Rasheed, T. Iliescu, and B. R. Noack. On closures for reduced order models $-$ A spectrum of first-principle to machine-learned avenues. *Physics of Fluids*, 33(9):091301, Sept. 2021. arXiv:2106.14954 [physics].

[2] F. Ancona, L. Caravenna, A. Cesaroni, G. M. Coclite, C. Marchi, and A. Marson. Analysis and control on networks: Trends and perspectives. *Networks & Heterogeneous Media*, 12(3), 2017.

[3] F. Ancona, A. Cesaroni, G. M. Coclite, and M. Garavello. On the optimization of conservation law models at a junction with inflow and flow distribution controls, July 2018.

[4] H. Attouch, G. Buttazzo, and G. Michaille. *Variational analysis in Sobolev and BV spaces: applications to PDEs and optimization*. SIAM, 2014.

[5] A. Aw and M. Rascle. Resurrection of "second order" models of traffic flow. *SIAM J. Appl. Math.*, 60:916–938, 2000.

[6] U. Baumgart and M. Burger. Optimal control of traffic flow based on reinforcement learning. In C. Klein, M. Jarke, M. Helfert, K. Berns, and O. Gusikhin, editors, *Smart Cities, Green Technologies, and Intelligent Transport Systems*, pages 313–329, Cham, 2022. Springer International Publishing.

[7] A. Bayen, M. L. Delle Monache, M. Garavello, P. Goatin, and B. Piccoli. *Control Problems for Conservation Laws with Traffic Applications: Modeling, Analysis, and Numerical Methods*, volume 99 of *Progress in Nonlinear Differential Equations and Their Applications*. Springer International Publishing, Cham, 2022.

[8] A. Bayen, R. Raffard, and C. Tomlin. Adjoint-based control of a new eulerian network model of air traffic flow. *IEEE Transactions on Control Systems Technology*, 14(5):804–818, Sept. 2006.

[9] H. Beirão da Veiga. Diffusion on viscous fluids. existence and asymptotic properties of solutions. *Annali della Scuola Normale Superiore di Pisa-Classe di Scienze*, 10(2):341–355, 1983.

[10] F. Berthelin, T. Goudon, and S. Minjeaud. Multifluid flows: A kinetic approach. *Journal of Scientific Computing*, 66(2):792–824, 2016.

[11] O. Biham, A. A. Middleton, and D. Levine. Self-organization and a dynamical transition in traffic-flow models. *Phys. Rev. A*, 46:R6124–R6127, Nov 1992.

[12] A. Braides. *Gamma-convergence for Beginners*, volume 22. Clarendon Press, 2002.

[13] A. Bressan. Hyperbolic systems of conservation laws. *Revista Matemática Complutense*, 12(1):135, Jan. 1999.

[14] G. Bretti, R. Natalini, and B. Piccoli. Numerical approximations of a traffic flow model on networks. *Networks and Heterogeneous Media*, 1(1):57, 2006.

[15] C. Calgaro, E. Creusé, and T. Goudon. Modeling and simulation of mixture flows: Application to powder–snow avalanches. *Computers & Fluids*, 107:100–122, 2015.

[16] A. Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical imaging and vision*, 20:89–97, 2004.

[17] Y. Chitour and B. Piccoli. Traffic circles and timing of traffic lights for cars flow. *Discrete Contin. Dyn. Syst. Ser. B*, 5:599–630, Aug. 2005.

[18] A. J. Chorin. The numerical solution of the navier-stokes equations for an incompressible fluid. *Bulletin of the American Mathematical Society*, 73(6):928–931, 1967.

[19] A. J. Chorin. Numerical solution of the navier-stokes equations. *Mathematics of computation*, 22(104):745–762, 1968.

[20] A. J. Chorin. On the convergence of discrete approximations to the navier-stokes equations. *Mathematics of computation*, 23(106):341–353, 1969.

[21] F. Clarelli, C. Di Russo, R. Natalini, and M. Ribot. Mathematical models for biofilms on the surface of monuments. In *Applied and industrial mathematics in Italy III*, pages 220–231. World Scientific, 2010.

[22] F. Clarelli, C. Di Russo, R. Natalini, and M. Ribot. A fluid dynamics model of the growth of phototrophic biofilms. *Journal of mathematical biology*, 66(7):1387–1408, 2013.

[23] G. M. Coclite, M. Garavello, and B. Piccoli. Traffic flow on a road network. *SIAM Journal on Mathematical Analysis*, 36(6):1862–1886, 2005.

[24] N. Cogan and J. P. Keener. The role of the biofilm matrix in structural development. *Mathematical Medicine and Biology*, (21):147–166, 2004.

[25] P. L. Combettes and J.-C. Pesquet. Proximal Splitting Methods in Signal Processing. In H. H. Bauschke, R. S. Burachik, P. L. Combettes, V. Elser, D. R. Luke, and H. Wolkowicz, editors, *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pages 185–212. Springer New York, New York, NY, 2011.

[26] V. Coscia, M. Delitala, and P. Frasca. On the mathematical theory of vehicular traffic flow ii discrete velocity kinetic models. *International Journal of NonLinear Mechanics*, 2007.

[27] C. Courtès, E. Franck, K. Lutz, L. Navoret, and Y. Privat. Reduced modelling and optimal control of epidemiological individual-based models with contact heterogeneity. *Optimal Control Applications and Methods*, 2023.

[28] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Systems*, 2, 1989.

[29] C. M. Dafermos. Polygonal approximations of solutions of the initial value problem for a conservation law. *ournal of Mathematical Analysis and Applications*, 38, 1972.

[30] C. D'Apice, R. Manzo, and B. Piccoli. Packet flow on telecommunication networks. *SIAM J. Math. Analysis*, 38:717–740, 01 2006.

[31] M. Delitala and A. Tosin. Mathematical modeling of vehicular traffic: a discrete kinetic theory approach. *Mathematical Models and Methods in Applied Sciences*, 17(06):901–932, 2007.

[32] S. V. Dongen. *Graph Clustering by Flow Simulation*. PhD thesis, Université d'Utrecht, Allemagne, 2000.

[33] L. D.R., G. P., and T. N.B. Contram: Structure of the model. *Transport and Road Research Laboratory (TRRL) Research Report 178, Department of Transport, Crowthorne*, 1989.

[34] D. Dutykh, C. Acary-Robert, and D. Bresch. Mathematical modeling of powder-snow avalanche flows. *Studies in Applied Mathematics*, 127(1):38–66, 2011.

[35] J. R. E. Candes and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. 2005.

[36] I. Ekeland and R. Témam. *Convex analysis and variational problems*, volume 28 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, english edition, 1999. Translated from the French.

[37] M. G. et al. Optimal control for traffic flow networks. *Journal of Optimization Theory and Applications*, 126, 2005.

[38] L. Fermo and A. Tosin. Fundamental diagrams for kinetic equations of traffic flow. *Discrete and Continuous Dynamical Systems - Series S*, 7:449–462, 2014.

[39] K.-I. Funahashi. On the approximate realization of continuous mappings by neural networks, neural networks. *Neural Network*, 2, 1989.

[40] A. Fügenschuh, M. Herty, A. Klar, and A. Martin. Combinatorial and Continuous Models for the Optimization of Traffic Flows on Networks. *SIAM Journal on Optimization*, 16(4):1155–1176, Jan. 2006.

[41] B. P. Gabriellle Bretti, Roberto Natalini. A fluid-dynamic traffic model on road networks. *Archives of Computational Methods in Engineering*, 2007.

[42] M. Garavello and B. Piccoli. *Traffic flow on networks: conservation law models*. Number Vol. 1 in AIMS series on applied mathematics. American Inst. of Mathematical Sciences, Springfield, Mo, 2006.

[43] P. Gipps. A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, 1981.

[44] P. Goatin, S. Göttlich, and O. Kolb. Speed limit and ramp meter control for traffic flow networks. *Engineering Optimization*, 48(7):1121–1144, July 2016.

[45] T. Goudon and A. Vasseur. On a model for mixture flows: Derivation, dissipation and stability properties. *Archive for Rational Mechanics and Analysis*, 220:1–35, 2016.

[46] S. Göttlich, M. Herty, and U. Ziegler. Modeling and optimizing traffic light settings in road networks. *Computers & Operations Research*, 55:36–51, Mar. 2015.

[47] K. Ito and S. S. Ravindran. A reduced-order method for simulation and control of fluid flows. *Journal of Computational Physic*, 143(2), 1998.

[48] R. Jayakrishnan, H. S. Mahmassani, and T.-Y. Hu. An evaluation tool for advanced traffic information and management systems in urban networks. *Transportation Research Part C: Emerging Technologies*, 1994.

[49] D. J.R. and X. Ru. The application of the method of quasi-reversibility to the sideways heat equation. *Journal of Mathematical Analysis and Applications*, 236, 1999.

[50] N. K. and S. M. A cellular automaton model for freeway traffic. *Journal de Physique I*, 1992.

[51] S. S. R. K. Ito. A reduced-order method for simulation and control of fluid flows. *Journal of Computational Physics*, 143(2), 1998.

[52] A. V. Kazhikhov and S. Smagulov. The correctness of boundary-value problems in a diffusion model of an inhomogeneous liquid. In *Soviet Physics Doklady*, volume 22, page 249, 1977.

[53] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[54] J. Krug and H. Spohn. Universality classes for deterministic surface growth. *PHYSICAL REVIEW A*, 1988.

[55] A. A. Kurzhanskiy and P. Varaiya. Active traffic management on road networks: a macroscopic approach. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 368(1928):4607–4626, Oct. 2010.

[56] B. S. L., N. B. R., and K. Petros. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52(1):477–508, 2020.

[57] S. Labarthe, B. Polizzi, T. Phan, T. Goudon, M. Ribot, and B. Laroche. A mathematical model to investigate the key drivers of the biogeography of the colon microbiota. *Journal of Theoretical Biology*, 462:552–581, 2019.

[58] E. B. Lee and L. Markus. *Foundations of optimal control theory [by] E. B. Lee [and] L. Markus*. Wiley New York, 1967.

[59] LeVeque and Randall. *Numerical methods for conservation laws*. Birkhäuser Verlag, 1992.

[60] T. Liard and E. Zuazua. Analysis and numerical solvability of backward-forward conservation laws. *SIAM Journal on Mathematical Analysis*, 55(3):1949–1968, 2023.

[61] M. Lighthill and G. Whitham. On kinematic waves - ii - a theory of traffic flow on long crowded roads. *Proc. R. Soc. Lond.*, 1955.

[62] T. M., H. A., and H. D. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E*, 2000.

[63] Z. H. M. A non-equilibrium traffic model devoid of gas-like behavior. *Transportation Research Part B: Methodological*, 36(3):275–290, March 2002.

[64] K. Malena, C. Link, L. Bußemas, S. Gausemeier, and A. Trächtler. Traffic estimation and mpc-based traffic light system control in realistic real-time traffic environments. In C. Klein, M. Jarke, M. Helfert, K. Berns, and O. Gusikhin, editors, *Smart Cities, Green Technologies, and Intelligent Transport Systems*, pages 232–254, Cham, 2022. Springer International Publishing.

[65] A. K. Martin Treiber. *Traffic Flow Dynamics*. Springer Berlin Heidelberg, 2013.

[66] T. Materne, A. Klar, M. Günther, and R. Wegener. An explicit kinetic model for traffic flow. In A. M. Anile, V. Capasso, and A. Greco, editors, *Progress in Industrial Mathematics at ECMI 2000*, pages 597–601, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[67] L. Michael, D. David, and P. J. M. Sparse mri: The application of compressed sensing for rapid mr imaging. *Magnetic Resonance in Medicine*, 58(6):1182–1195, 2007.

[68] L. Michael, D. D. L., S. J. M., and J. M. Pauly. Compressed sensing mri. *IEEE Signal Processing Magazine*, 25(2):72–82, 2008.

[69] B. Nathan, A. Frank, B. Timo, H. Aric, K. Yannis, N. Habib, P. Manish, P. Abani, S. James, W. Stefan, W. Karen, and S. Lee. Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence. 2 2019.

[70] B. R. Noack, P. Papas, and P. A. Monkewitz. The need for a pressure-term representation in empirical Galerkin models of incompressible shear flows. *Journal of Fluid Mechanics*, 523:339–365, Jan. 2005.

[71] S. L. Paveri-Fontana. On boltzmann-like treatments for traffic flow: A critical review of the basic model and an alternative proposal for dilute traffic analysis. *Transportation Research*, 9:225–235, 1975.

[72] H. J. Payne. Model of freeway traffic and control. *Mathematical Model of Public System*, pages 51–61, 1971.

[73] B. W. Peterson, Y. He, Y. Ren, A. Zerdoum, M. R. Libera, P. K. Sharma, A.-J. van Winkelhoff, D. Neut, P. Stoodley, H. C. van der Mei, and H. J. Busscher. Viscoelasticity of biofilms and their recalcitrance to mechanical and chemical challenges. *FEMS microbiology reviews*, 39(2):234–245, 03 2015.

[74] B. Polizzi, O. Bernard, and M. Ribot. A time-space model for the growth of microalgae biofilms for biofuel production. *Journal of Theoretical Biology*, 432:55 – 79, 2017.

[75] B. Polizzi, A. Fanesi, F. Lopes, M. Ribot, and O. Bernard. Understanding photosynthetic biofilm productivity and structure through 2d simulation. *PLOS Computational Biology*, 18(4):1–20, 04 2022.

[76] L. Preziosi and A. Tosin. Multiphase modelling of tumour growth and extracellular matrix interaction: mathematical tools and applications. *Journal of Mathematical Biology*, 58(4):625, Oct 2008.

[77] I. Prigogine, P. Resibois, R. Herman, and R. Anderson. On a generalized boltzmann-like approach for traffic flow. *Bulletin de la Classe des sciences. Académie royale de Belgique*, 48(9):805–814, 1962.

[78] S. C. B. P. J.-M. Qiu and T. Ren. Runge-kutta discontinuous galerkin method for traffic flow model on networks. *Journal of Scientific Computing*, 2015.

[79] W. R. Simulation des strassenverkehrsflusses. *Institute for Traffic Engineering, University of Karlsruhe, Tech. Rep*, 1974.

[80] K. Rajagopal and L. Tao. *Mechanics of mixtures*. Series on Advances in Mathematics for Applied Sciences. World Scientific, Singapore, 1995.

[81] J. Reilly, W. Krichene, M. L. D. Monache, S. Samaranayake, P. Goatin, and A. Bayen. Adjoint-based optimization on a network of discretized scalar conservation law PDEs with applications to coordinated ramp metering.

[82] P. I. Richards. Shock waves on the highway. *Operations Research*, 1956.

[83] H. E. Robbins. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[84] H. E. Robbins. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[85] D. Ryckelynck. A priori hyperreduction method: an adaptive approach. *Journal of Computational Physics*, 202, 2005.

[86] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry. How Does Batch Normalization Help Optimization?, Apr. 2019. arXiv:1805.11604 [cs, stat].

[87] P. J. Schnurr and D. G. Allen. Factors affecting algae biofilm growth and lipid production: A review. *Renewable and Sustainable Energy Reviews*, 52:418–429, 2015.

[88] P. Secchi. On the motion of viscous fluids in the presence of diffusion. *SIAM Journal on Mathematical Analysis*, 19(1):22–31, 1988.

[89] W. Snyder, C. Mou, H. Liu, O. San, R. De Vita, and T. Iliescu. Reduced Order Model Closures: A Brief Tutorial, Feb. 2022. arXiv:2202.14017 [physics].

[90] R. Temam. Sur l'approximation de la solution des équations de navier-stokes par la méthode des pas fractionnaires (i). *Archive for Rational Mechanics and Analysis*, 32:135–153, 1969.

[91] C. Truesdell. Sulle basi della termomeccanica. I, II. *Atti Accad. Naz. Lincei. Rend. Cl. Sci. Fis. Mat. Nat. (8)*, 1957.

[92] C. Truesdell. *Rational thermodynamics*. McGraw-Hill Book Co., New York-London-Sydney, 1969. A course of lectures on selected topics, With an appendix on the symmetry of the heat-conduction tensor by C. C. Wang.

[93] C. Truesdell and R. Toupin. The classical field theories. In *Handbuch der Physik, Bd. III/1*. Springer, Berlin, 1960. With an appendix on tensor fields by J. L. Ericksen.

[94] R. Wegener and A. Klar. A kinetic model for vehicular traffic derived from a stochastic microscopic model. *Transport Theory and Statistical Physics*, 25(7):785–798, 1996.

[95] G. B. Whitham. *Linear and nonlinear waves*. John Wiley & Sons, 2011.

[96] C. William, Z. Yudong, P. B. S., J. Genlin, and D. Zhengchao. Energy preserved sampling for compressed sensing mri. *Computational and Mathematical Methods in Medicine*, 2014.

[97] Y. Wu, L. Liu, J. Bae, K.-H. Chow, A. Iyengar, C. Pu, W. Wei, L. Yu, and Q. Zhang. Demystifying Learning Rate Policies for High Accuracy Training of Deep Neural Networks, Oct. 2019. arXiv:1908.06477 [cs, stat].

[98] Y. G. YuFeng Shi. A maximum-principle-satisfying finite volume compact-weno scheme for traffic flow model on networks. *Applied Numerical Mathematics*, 108:21–36, 2016.

Comment optimiser l'évacuation en cas de crise et l'intervention des services d'urgence sur les réseaux routiers ? Comment simuler rapidement la dynamique du trafic routier avec des modèles réduits ? Cette thèse aborde ces questions en utilisant des approches de contrôle optimal et d'intelligence artificielle. Nous modélisons la première question à l'aide d'un problème de contrôle visant à libérer un axe routier de façon optimale, en utilisant la signalisation et des barrages. Notre méthode utilise une semi-discrétisation de l'équation de Lighthill, Whitham et Richards (LWR) sur un graphe de réseau routier. Nous introduisons des fonctions de commande à chaque jonctions pour traduire une action de délestage de l'axe de circulation. Dans un souci d'implémentation pratique, nous prenons également en compte la contrainte d'obtenir des contrôles parcimonieux. De façon pratique, un tel problème est complexe à mettre en œuvre numériquement, en raison de sa dimension, du caractère fortement non-linéaire du critère et du nombre important de minima locaux. Nous introduisons et testons un algorithme numérique reposant sur l'analyse des conditions d'optimalité d'optimalité de ce problème. Un code de résolution dans le langage Julia est mis à disposition en open source. Dans une seconde partie, nous cherchons à proposer des modèles réduits de trafic afin de faciliter les simulations numériques de réseaux routiers. Nous cherchons à préserver la fiabilité du modèle tout en réduisant son ordre en utilisant des bases réduites associées à des techniques d'apprentissage profond.

How to optimize crisis evacuation and emergency service response on road networks? How can road traffic dynamics be rapidly simulated using scale models? This thesis addresses these questions using optimal control and artificial intelligence approaches. We model the first question using a control problem aimed at optimally clearing a road axis, using signalling and barricades. Our method uses a semi-discretization of the Lighthill, Whitham and Richards (LWR) equation on a road network graph. We introduce control functions at each junction to reflect a relieving action on the traffic axis. For the sake of practical implementation, we also take into account the constraint of obtaining sparse controls. In practice, such a problem is difficult to implement numerically, due to its size, the highly non-linear nature of the criterion and the large number of local minima. We introduce and test a numerical algorithm based on the analysis of optimality conditions for this problem. A solver code in the Julia language is made available as open source. In a second part, we propose reduced traffic models to ease numerical simulations of road networks. We aim to preserve the reliability of the model while reducing its order by using reduced bases associated with deep learning techniques.

INSTITUT DE RECHERCHE MATHÉMATIQUE AVANCÉE
UMR 7501
Université de Strasbourg et CNRS
7 Rue René Descartes
67 084 STRASBOURG CEDEX

Tél. 03 68 85 01 29
Fax 03 68 85 03 28

https://irma.math.unistra.fr
irma@math.unistra.fr

Université de Strasbourg

Institut de Recherche Mathématique Avancée

IRMA 2023/11

ISSN 0755-3390

telnum