

RAPPORT DE STAGE

---

**Simulation 2D de l'équation du transfert radiatif  
et reconstruction de la densité par un réseau de  
neurones**

---

*Stagiaire*  
Roussel Desmond NZOYEM

*Maîtres de stage*  
Emmanuel FRANCK  
Laurent NAVORET  
Vincent VIGON

*Enseignant référent*  
Christophe PRUD'HOMME

*Stage réalisé dans le cadre du master 1 CSMI  
du 15 juin au 15 août 2020  
à l'initiative de l'équipe MOCO  
au sein du service IRMA à l'UFR Math-Info de l'Unistra*

Année académique 2019 - 2020

21 août 2020

## *Remerciements*

Je tiens à remercier dans un premier temps mes maîtres de stage MM. Emmanuel FRANCK, Laurent NAVORET, et Vincent VIGON de m'avoir permis d'effectuer un stage scientifique très enrichissant dans les meilleures conditions possibles. Étant donné la situation sanitaire de COVID-19, ils ont su me transmettre tous les enseignements nécessaires pour effectuer mes différentes missions (et bien plus encore), à distance comme en présentiel.

J'étends mes remerciements à M. Christophe PRUD'HOMME dont l'aide a été précieuse dans le rétablissement des ressources de calcul nécessaires pour les tâches de simulation.

Je remercie aussi mes camarades de CSMI Guillaume STEIMER et Léo BOIS pour leurs conseils. Les outils et méthodes de travail qu'ils ont recommandé (format binaire, clarification des paramètres, etc.) m'ont permis d'augmenter ma vitesse de travail et d'améliorer mes résultats.

# Table des matières

<b>Remerciements</b>	<b>ii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Présentation de l'IRMA</b>	<b>2</b>
2.1 Structure . . . . .	2
2.2 Les équipes . . . . .	2
<b>3 Simulation 2D</b>	<b>4</b>
3.1 Le transfert radiatif . . . . .	4
3.2 Schéma de splitting . . . . .	6
3.2.1 Étape 1 . . . . .	6
3.2.2 Étape 2 . . . . .	7
3.3 Implémentation en C++ . . . . .	9
3.3.1 Configuration de la simulation . . . . .	9
3.3.2 Sauvegarde des données . . . . .	10
3.4 Résultats . . . . .	10
<b>4 Apprentissage</b>	<b>14</b>
4.1 Description des entrées/sorties . . . . .	14
4.1.1 En 1D . . . . .	15
4.1.2 En 2D . . . . .	16
4.2 Architecture générale . . . . .	17
4.3 Les couches utilisées . . . . .	18
4.3.1 Les couches de convolution . . . . .	18
4.3.2 Le Max-pooling . . . . .	20
4.3.3 Flatten . . . . .	20
4.3.4 Les couches denses . . . . .	21
4.4 Configuration de l'entraînement . . . . .	21
4.4.1 Les métriques . . . . .	21
4.4.2 Les hyper-paramètres . . . . .	22
4.5 Résultats . . . . .	23
4.5.1 Régression . . . . .	24
4.5.2 Classification . . . . .	28
4.6 Conclusion sur l'apprentissage . . . . .	29
<b>5 Bilan du stage</b>	<b>31</b>
5.1 Ressources utilisées . . . . .	31
5.1.1 Édition de code . . . . .	31
5.1.2 Simulations . . . . .	31
5.1.3 Apprentissage . . . . .	31
5.1.4 Communication . . . . .	31

5.2	Journal de bord . . . . .	32
5.2.1	Semaines 1 et 2 . . . . .	32
5.2.2	Semaines 3 et 4 . . . . .	32
5.2.3	Semaines 5 et 6 . . . . .	32
5.2.4	Semaines 7, 8, et 9 . . . . .	32
5.3	Difficultés rencontrées et solutions apportées . . . . .	33
5.3.1	Apparition d'un créneau indésirable . . . . .	33
5.3.2	Détection de la position du créneau . . . . .	33
5.3.3	Gestion du temps . . . . .	34
5.4	Les apports du stage . . . . .	34
5.4.1	Expérience en développement . . . . .	34
5.4.2	Équations aux dérivées partielles . . . . .	34
5.4.3	Réseaux de neurones . . . . .	34
5.4.4	Expérience de recherche . . . . .	34
<b>6</b>	<b>Conclusion</b>	<b>35</b>
<b>A</b>	<b>Comment reproduire les resultats ?</b>	<b>36</b>
A.1	Exécution du code 1D/2D . . . . .	36
A.2	Lecture du format binaire . . . . .	36
A.3	Exécution des notebooks . . . . .	36
<b>B</b>	<b>Comment faire des prédictions avec ce modèle ?</b>	<b>37</b>
B.1	Normalisation des données . . . . .	37
B.2	Chargement du modèle . . . . .	37
B.3	Entraîner le modelé en continu . . . . .	37
	<b>Bibliographie</b>	<b>38</b>

# Liste des symboles

Symbole	Définition	Unité
$x$	Abscisse	cm
$y$	Ordonnée	cm
$\mathbf{x}$	Vecteur position	cm
$\Omega$	Vecteur direction de propagation des photons	$\text{cm}^2/\text{cm}^2$
$\nu$	Fréquence	Hz
$p$	Fonction de distribution angulaire de « scattering »	
$I$	Intensité spécifique de radiation	$\text{W}/\text{m}^2/\text{sr}/\text{Hz}$
$B$	Fonction de Planck	$\text{W}/\text{m}^2/\text{sr}/\text{Hz}$
$\rho$	Densité du milieu	$\text{g cm}^{-3}$
$\sigma_a$	Opacité d'absorption	$\text{cm}^{-1}$
$\sigma_c$	Opacité de « scattering » (ou de dispersion)	$\text{cm}^{-1}$
$t$	Temps	sh
$C_v$	Capacité thermique du milieu	Jerk/g/keV
$a$	Constante radiative (ou de rayonnement)	$\text{g}/\text{cm}/\text{sh}^2/\text{keV}$
$c$	Vitesse de la lumière	cm/sh
$T$	Température matière	keV
$E$	Energie des photons	$\text{g}/\text{cm}/\text{sh}^2$
$F$	Flux de photons	$\text{g}/\text{sh}^2$



## Chapitre 1

# Introduction

En 2015, le réseau de neurones vainqueur de l'ILSVRC<sup>1</sup> obtient une précision de 97.3 %, ce qui conduit les chercheurs à affirmer que les machines peuvent identifier les objets dans des images mieux que les humains (MARKOFF, 2015). Depuis lors, le domaine du Machine Learning a continué à prendre de l'ampleur. Aujourd'hui ses applications se multiplient dans plusieurs secteurs d'activité parmi lesquelles l'automobile, la finance, le divertissement, et plus important, celui de la santé à travers l'imagerie médicale.

Les tumeurs ont des propriétés optiques différentes des tissus qui les entourent<sup>2</sup>. Étant donné un domaine avec un faisceau lumineux qui s'y propage, reconstruire sa densité à l'aide du signal temporel mesuré sur ses bords constitue un problème inverse. Les problèmes inverses sont très importants en sciences mathématiques et ont des applications variées en imagerie médicale, radar, vision, etc. Ils sont malheureusement très difficiles à résoudre et nécessitent des algorithmes d'optimisation très avancés. Les réseaux de neurones artificiels se présentent comme une méthode potentiellement moins coûteuse mais plus rapide.

Grace à son unité mixte de recherche IRMA, l'UFR de mathématiques et d'informatique de l'Université de Strasbourg est un pôle de recherche en mathématiques appliquées. À travers ses équipes MOCO et Probabilités, l'IRMA s'intéresse aux interactions entre les EDP et le Machine Learning, raison pour laquelle j'ai choisi d'y effectuer mon stage de master 1 CSMI<sup>3</sup>. Au cours de ce stage (du 15 juin au 15 août 2020), j'ai pu m'intéresser au problème inverse de reconstruction de la densité d'un domaine par un réseau de neurones convolutif (CNN).

Ce stage a été suivi par les enseignants-chercheurs MM. Emmanuel FRANCK, Laurent NAVO-RET, et Vincent VIGON et s'inscrit dans la continuation d'un projet (encadré par la même équipe) qui s'est déroulé du 19 mars au 28 mai 2020. Le projet consistait en la simulation 1D d'un schéma de « splitting » pour le modèle P1 de l'équation du transfert radiatif couplé avec la matière. Le stage quant à lui a essentiellement consisté en la simulation du même schéma en 2D, et en la reconstruction de la densité du milieu par un CNN. Ce stage a été l'opportunité pour moi d'apprendre sur les EDP et sur l'apprentissage profond tout en me familiarisant avec l'interface de programmation de la librairie de réseaux de neurones Keras.

En vue de rendre compte de manière fidèle des deux mois passés au sein de l'IRMA, il apparaît logique de présenter en titre de préambule le cadre du stage et son environnement technique. Ensuite il s'agira de présenter les différentes missions et tâches qui j'ai pu effectuer. Enfin je présenterais un bilan du stage, en incluant les différents apports et enseignements que j'ai pu en tirer.

---

1. ImageNet Large Scale Visual Recognition Challenge

2. Les tumeurs sont généralement plus denses que les tissus sains.

3. Calcul Scientifique et Mathématiques de l'Information

## Chapitre 2

# Présentation de l'IRMA

*Les informations présentées dans cette section sont entièrement issues du [site web de l'IRMA](#).*

Créée en 1966 par Jean FRENJEL et Georges REEB, l'IRMA<sup>1</sup> est une unité mixte de recherche (UMR 7501) sous la double tutelle du CNRS (à travers l'INSMI<sup>2</sup>) et de l'Université de Strasbourg (à travers l'UFR de mathématiques et d'informatique).

Dirigée par le professeur Philippe HELLUY, l'IRMA comporte environ 130 membres. On y compte environ 87 chercheurs et enseignants-chercheurs permanents et une quarantaine de membres non permanents repartis en 7 équipes de recherche.

Les activités majeures de l'entreprise sont l'organisation des séminaires, des journées, des colloques et des conférences. Ces activités sont renforcées par les nombreux partenariats qu'elle maintient dans les secteurs académique (Cemosis, Labex IRMIA, etc.) et industriel (AxesSIM, Electis, etc.).

## 2.1 Structure

L'organigramme de l'entreprise est représenté à la figure 2.1.

## 2.2 Les équipes

L'équipe MOCO<sup>3</sup> se compose de spécialistes des EDP, de la théorie du contrôle, du calcul scientifique, du calcul haute performance et des statistiques. Les enseignants-chercheurs MM. Emmanuel FRANCK et Laurent NAVORET y sont responsables des séminaires en équations aux dérivées partielles.

L'équipe Probabilités est composée d'experts en statistique et calcul de probabilités. Ses membres se retrouvent régulièrement lors du Séminaire Stochastique. C'est à cette équipe qu'appartient M. Vincent VIGON.

Je tiens une fois de plus à remercier les trois chercheurs mentionnés ci-hauts qui ont encadré ce stage. La combinaison des deux équipes dont ils font partie a permis de faire face aux deux aspects de ce stage : premièrement la simulation d'EDP et deuxièmement l'utilisation des réseaux de neurones.

---

1. Institut de Recherche Mathématique Avancée

2. Institut National des Sciences Mathématiques et de leurs Interactions

3. MOdélisation et COntôle



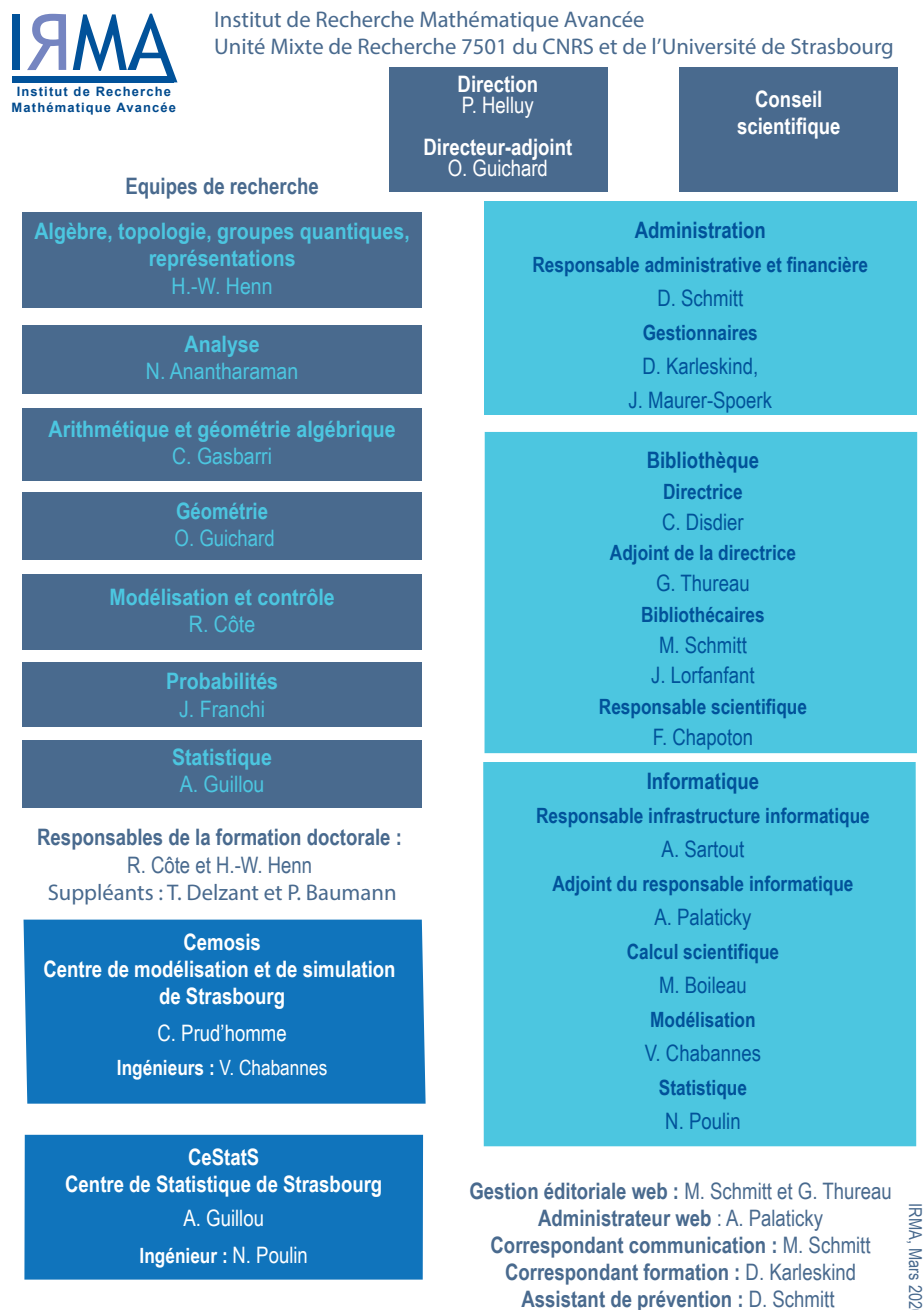


FIGURE 2.1 – Organigramme représentant la structure de l'IRMA au mois de mars 2020 (IRMA, 2020).

## Chapitre 3

# Simulation 2D

Ayant simulé le schéma de « splitting » en 1D avant le début du stage, on s'intéresse dans cette partie à sa simulation en 2D. Il s'agit donc de résoudre le problème direct du transfert radiatif avant de passer au problème inverse dans le chapitre suivant. On rappelle brièvement l'équation du transfert radiatif (ETR), le modèle P1 couplé avec la matière, le schéma de « splitting », avant de présenter l'implémentation en C++.

### 3.1 Le transfert radiatif

On considère un rayonnement transporté par des particules de masse nulle appelés photons. Lorsqu'ils se trouvent en présence de la matière, les photons interagissent avec les atomes. Trois phénomènes sont prépondérants (figure 3.1) :

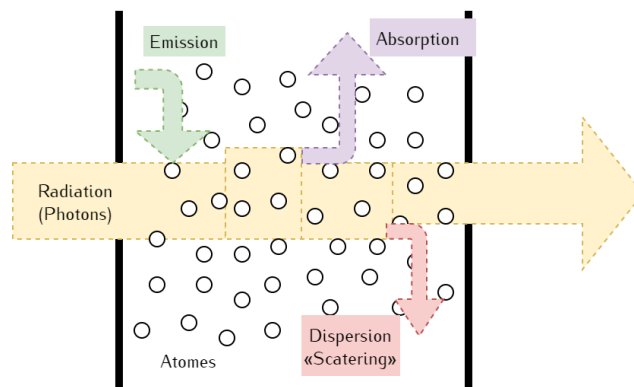


FIGURE 3.1 – Illustration des interactions entre radiation et matière.

- L'**émission** : des photons sont émis en réponse aux électrons excités descendants à des niveaux d'énergie plus bas. Ce phénomène est caractérisé par l'opacité d'émission  $\sigma_e$ . Il s'agit de l'inverse du libre parcours moyen d'émission<sup>1</sup>. Plus la température matière est élevée, plus ce phénomène est important.
- L'**absorption** : à l'inverse, certains photons sont absorbés, les électrons deviennent plus excités (ou se libèrent complètement de leurs atomes), et la matière se réchauffe. Ce phénomène se caractérise par l'opacité d'absorption  $\sigma_a$ . Lorsqu'on est à l'équilibre thermique,  $\sigma_a = \sigma_e$ .
- La **dispersion** (ou « **scattering** » ou parfois **diffusion**) : certains photons sont déviés de leur trajectoire originale par la matière. Ce phénomène se caractérise non seulement par son

1. Le libre parcours moyen d'émission représente la distance moyenne entre deux émissions consécutives de photons. Les libres parcours moyen d'absorption et de dispersion sont définis de manière similaire.

opacité de « scattering »  $\sigma_c$ <sup>2</sup>, mais aussi par une fonction de distribution angulaire décrivant la manière dont les photons sont déviés (TURPAULT, 2004, p. 13).

À un instant  $t$ , il faut 6 variables pour décrire la probabilité de présence d'un photon à une position  $\mathbf{x}$  donnée, ayant un moment donné. Pour décrire le moment, on utilise par convention la fréquence  $\nu$  et la direction de propagation du photon  $\Omega$  (pour laquelle il faut deux variables angulaires). Avec ces variables, on peut définir l'intensité radiative spécifique  $I(t, \mathbf{x}, \Omega, \nu)$  qui est directement proportionnelle au nombre de photons se trouvant à l'instant  $t$  dans le volume  $d\mathbf{x}$  autour du point  $\mathbf{x}$ , avec une fréquence  $\nu$  dans un intervalle fréquentiel de longueur  $d\nu$  et se propageant autour d'une direction  $\Omega$  dans un angle solide  $d\Omega$ . On écrit :

$$I(t, \mathbf{x}, \Omega, \nu) = ch\nu f(t, \mathbf{x}, \Omega, \nu)$$

Où  $f$  est la fonction de distribution des photons et  $h$  la constante de Planck (TURPAULT, 2004, p. 14).

L'équation du transfert radiatif (ETR) (3.1) représente un bilan d'énergie lié au rayonnement au niveau mésoscopique. Nous nous placerons dans le cas particulier de l'équilibre thermodynamique local (ETL)<sup>3</sup>. L'équilibre radiatif<sup>4</sup> quant à lui sera considéré plus tard comme condition initiale pour les simulations.

$$\begin{aligned} \frac{1}{c} \frac{\partial}{\partial t} I(t, \mathbf{x}, \Omega, \nu) + \Omega \cdot \nabla_{\mathbf{x}} I(t, \mathbf{x}, \Omega, \nu) &= \sigma_a(\rho, \Omega, \nu) (B(\nu, T) - I(t, \mathbf{x}, \Omega, \nu)) \\ &+ \frac{1}{4\pi} \int_0^\infty \int_{S^2} \sigma_c(\rho, \Omega, \nu) p(\Omega' \rightarrow \Omega) (I(t, \mathbf{x}, \Omega', \nu) - I(t, \mathbf{x}, \Omega, \nu)) d\Omega' d\nu \end{aligned} \quad (3.1)$$

Où  $p$  est la fonction de distribution angulaire de « scattering » caractérisée par la relation :

$$\oint p(\Omega' \rightarrow \Omega) d\Omega' = 1$$

$B(\nu, T)$  est la fonction de Planck. Il s'agit du champ radiatif (isotrope et homogène) qui coexiste avec la matière lors d'un équilibre thermique. Elle est donnée par :

$$B(\nu, T) = \frac{2h\nu^3}{c^2} \left[ \exp\left(\frac{h\nu}{kT}\right) - 1 \right]^{-1}$$

Où  $k$  est la constante de Boltzmann. Les autres termes sont définis dans la liste des symboles de la page 1.

Plusieurs modèles existent pour modéliser l'ETR avec différents niveaux de précision. Le modèle P1 est un modèle macroscopique<sup>5</sup> aux moments (d'ordre 2), linéaire et hyperbolique. Vu que l'énergie du rayonnement n'est pas conservée durant son interaction avec la matière, il faut coupler le modèle P1 avec une équation régissant l'énergie de la matière. On utilisera une équation d'énergie matière simplifiée qui ne tient compte que des termes d'échange avec le rayonnement. Le modèle P1 couplé avec la matière est présenté ci-bas (FRANCK, 2012) :

$$\begin{cases} \partial_t E + c \operatorname{div} \mathbf{F} = c\sigma_a (aT^4 - E) \\ \partial_t \mathbf{F} + c \nabla E = -c\sigma_c \mathbf{F} \\ \rho C_v \partial_t T = c\sigma_a (E - aT^4) \end{cases} \quad (3.2)$$

2.  $\sigma_a$  et  $\sigma_c$  sont définis de manière similaire à  $\sigma_e$

3. État dans lequel on peut définir une température pour chaque point du domaine, et l'émission est décrite par la fonction de Planck (TURPAULT, 2004).

4. Il se produit si la matière est à l'équilibre avec le rayonnement. Si on est dans l'ETL, les photons sont émis suivant la fonction de Planck à la température de la manière.

5. Il ne prend en compte que les variables d'espace et de temps et est obtenu par intégration de termes mésoscopiques (tels que  $I$ ) par rapport à la fréquence et la direction.

Où :

- $T(t, \mathbf{x}) > 0$  est la température du milieu
- $E(t, \mathbf{x}) \in \mathbb{R}$  est l'énergie des photons
- $\mathbf{F}(t, \mathbf{x}) \in \mathbb{R}^n$  est le flux des photons ( $n$  étant la dimension du problème)
- $\rho(\mathbf{x}) > 0$  est la densité du milieu
- $\sigma_a(\rho, T) > 0$  l'opacité d'absorption du milieu
- $\sigma_c(\rho, T) > 0$  l'opacité de « scattering » du milieu<sup>6</sup>

Partant de l'ETR (équation 3.1),  $E$  et  $\mathbf{F}$  sont définis de la manière suivante :

$$E(t, \mathbf{x}) = \frac{4\pi}{c} \int_0^\infty \int_{S^2} I(t, \mathbf{x}, \boldsymbol{\Omega}, \nu) d\boldsymbol{\Omega} d\nu$$

$$\mathbf{F}(t, \mathbf{x}) = \frac{4\pi}{c} \int_0^\infty \int_{S^2} \boldsymbol{\Omega} I(t, \mathbf{x}, \boldsymbol{\Omega}, \nu) d\boldsymbol{\Omega} d\nu$$

Comme on peut le voir à travers cette définition de  $E$  et  $\mathbf{F}$ , notre modèle est dit "gris" car nous l'intégrons sur tout le spectre de fréquence. En effet, nous nous intéressons au rayonnement à travers son bilan d'énergie transporté par le flux radiatif. Sur ce point, la version du modèle P1 que nous avons utilisé est moins précise qu'un modèle mésoscopique résolu par une méthode de Monte-Carlo ou une méthode des ordonnées discrètes. Néanmoins ce modèle P1 présente l'avantage d'être très peu coûteux et relativement facile à implémenter (TURPAULT, 2004, p. 18).

## 3.2 Schéma de splitting

Le modèle P1 tend vers une équation de diffusion lorsque les opacités d'absorption ( $\sigma_a$ ) et de dispersion ( $\sigma_c$ ) sont élevées (de façon à ce que  $c/\sigma_a \approx 1$ ). Ce cas limite est appelé **approximation de diffusion** (équation 3.3). Un schéma pour l'ETR qui préserve ce régime est dit « asymptotic preserving ».

$$\partial_t (aT^4 + \rho C_v T) - \operatorname{div} \left( \frac{acT^3}{\sigma_a} \nabla T \right) = O \left( \frac{1}{c} \right) \quad (3.3)$$

Les schémas classiques tels que le schéma de Rusanov ne sont pas assez précis pour capturer cette propriété. Le schéma en deux étapes (ou de « splitting ») proposé par M. FRANCK est « asymptotic preserving » et permet de pallier ce problème. Ses deux étapes sont résumées ci-bas.

### 3.2.1 Étape 1

La première étape (dite étape de **couplage** ou d'**équilibre**, ou de **relaxation de la température**) permet de régler la température sur chaque maille (indépendamment des autres mailles). On ne considère que les équations où la température est impliquée (EDP 1 et 3 du modèle P1, voir équations 3.4), en fixant la valeur du flux sur chaque maille. Il s'agit d'une méthode de point fixe qui est toujours définie (FRANCK, 2012, p. 161).

$$\begin{cases} \partial_t E + c \operatorname{div} \mathbf{F} = c\sigma_a (aT^4 - E) \\ \rho C_v \partial_t T = c\sigma_a (E - aT^4) \end{cases} \quad (3.4)$$

Le domaine rectangulaire est supposé discrétisé en  $N \times M$  mailles uniformes ; on se trouve sur la maille  $j$  (voir figure 3.2a pour les détails de la discrétisation) à l'étape d'itération  $n$ . On pose

---

6.  $\sigma_a$  et  $\sigma_c$  ne dépendent que de  $\rho$  et  $T$  si on se place dans l'ETL.

donc  $\Theta = aT^4$  et on obtient le système :

$$\begin{cases} \frac{E_j^{q+1} - E_j^n}{\Delta t} = c\sigma_a(\Theta_j^{q+1} - E_j^{q+1}) \\ \rho_j C_v \mu_q \frac{\Theta_j^{q+1} - \Theta_j^n}{\Delta t} = c\sigma_a(E_j^{q+1} - \Theta_j^{q+1}) \end{cases}$$

Où  $\mu_q = \frac{1}{T^{3,n} + T^n T^{2,q} + T^q T^{2,n} + T^{3,q}}$ . L'étape revient à résoudre un système de Cramer. On obtient donc :

$$\begin{cases} E_j^{q+1} = \frac{\alpha E_j^n + \beta \gamma \Theta_j^n}{1 - \beta \delta} \\ \Theta_j^{q+1} = \frac{\gamma \Theta_j^n + \alpha \delta E_j^n}{1 - \beta \delta} \end{cases} \quad (3.5)$$

Avec  $\alpha = \frac{1}{\Delta t \left( \frac{1}{\Delta t} + c\sigma_a \right)}$ ,  $\beta = \frac{c\sigma_a}{\frac{1}{\Delta t} + c\sigma_a}$ ,  $\gamma = \frac{\rho_j C_v \mu_q}{\Delta t \left( \frac{\rho_j C_v \mu_q}{\Delta t} + c\sigma_a \right)}$  et  $\delta = \frac{c\sigma_a}{\frac{\rho_j C_v \mu_q}{\Delta t} + c\sigma_a}$ .

On itère ainsi sur  $q$  jusqu'à ce que  $E$  et  $\Theta$  convergent respectivement vers  $E^*$  et  $\Theta^*$ .  $F$  reste inchangé durant cette étape.

### 3.2.2 Étape 2

Il s'agit ici de résoudre les deux premières EDP du modèle P1 (système d'équation 3.2). Avant d'attaquer le schéma de « splitting », on note que les équations à résoudre (3.6) sont hyperboliques et que la méthode des volumes finis est donc la mieux adaptée.

$$\begin{cases} \partial_t E + c \operatorname{div} \mathbf{F} = c\sigma_a (aT^4 - E) \\ \partial_t \mathbf{F} + c \nabla E = -c\sigma_c \mathbf{F} \end{cases} \quad (3.6)$$

On rappelle que la température a été réglée à l'étape 1 du schéma (i.e on ne considère plus le terme  $E - aT^4$ ). On se place sur une maille  $j$  caractérisée par son volume  $\Omega_j$ <sup>7</sup>.

$$\begin{cases} \partial_t \int_{\Omega_j} E + c \int_{\Omega_j} \operatorname{div} \mathbf{F} = 0 \\ \partial_t \int_{\Omega_j} \mathbf{F} + c \int_{\Omega_j} \nabla E = -c\sigma_c \int_{\Omega_j} \mathbf{F} \end{cases}$$

On définit une normale  $\mathbf{n}_j$ <sup>8</sup> à la surface  $\Omega_j$  et on applique le théorème de la divergence. On moyenne ensuite les intégrales sur chaque maille pour obtenir :

$$\begin{cases} \partial_t E_j + \frac{c}{|\Omega_j|} \int_{\partial\Omega_j} (\mathbf{F}, \mathbf{n}_j) = 0 \\ \partial_t \mathbf{F}_j + \frac{c}{|\Omega_j|} \int_{\partial\Omega_j} E \mathbf{n}_j = \frac{c\sigma_c}{|\Omega_j|} \int_{\Omega_j} \mathbf{F} \end{cases} \quad (3.7)$$

Où

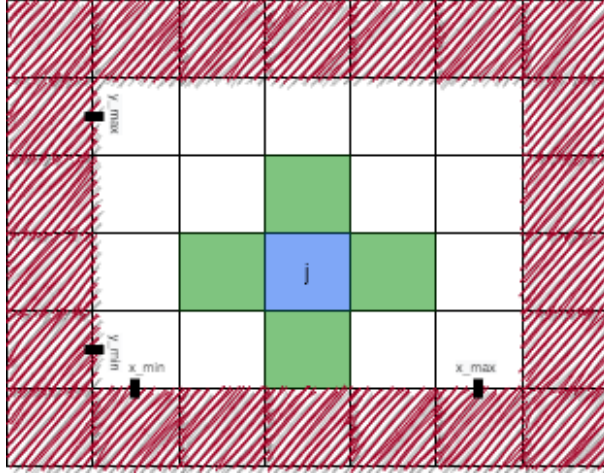
$$E_j(t) = \frac{1}{|\Omega_j|} \int_{\Omega_j} E(t, \mathbf{x}) \quad \text{et} \quad \mathbf{F}_j(t) = \frac{1}{|\Omega_j|} \int_{\Omega_j} \mathbf{F}(t, \mathbf{x})$$

7. On utilisera généralement le terme "maille" vu qu'en 2D, le "volume"  $\Omega_j$  est en réalité une surface.

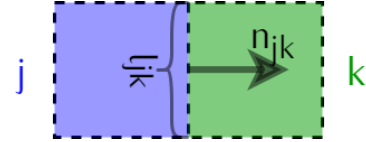
8. Il s'agit en réalité d'une série de quatre normales définies par rapport aux mailles voisines de  $j$  (voir figure 3.2b).

Et  $(\cdot, \cdot)$  désigne un produit scalaire dans  $\mathbb{R}^n$  ( $n$  étant la dimension du problème).

Nous devons maintenant retourner sur le maillage en discrétisant les intégrales sur le bord  $\partial\Omega_j$  de la maille  $j$ . Durant cette étape, il faut considérer l'ajout de mailles "fantômes" (figure 3.2a), ce qui porte le nombre total de mailles à  $(N + 2) \times (M + 2)$ .



(A) Maillage 2D



(B) Interaction entre deux mailles  $j$  et  $k$

FIGURE 3.2 – Discrétisation du maillage 2D. Sur la figure (A), on peut observer les mailles dites "fantômes" hachurées en rouge. Les quatre mailles voisines d'une maille  $j$  sont indiquées en vert. Les nombres de mailles suivant la verticale  $N$  et suivant l'horizontale  $M$  sont choisis telle que le maillage soit uniforme i.e  $\Delta x = \frac{x_{\max} - x_{\min}}{N} = \frac{y_{\max} - y_{\min}}{M} = \Delta y$ . Sur la figure (B), on observe la représentation d'une des normales  $\mathbf{n}_{jk}$  de la mailles  $j$ . On peut aussi observer la longueur du bord commun  $l_{jk}$ .

On obtient donc le système :

$$\begin{cases} \partial_t E_j + \frac{c}{|\Omega_j|} \sum_k l_{jk} (\mathbf{F}_{jk}, \mathbf{n}_{jk}) = 0 \\ \partial_t \mathbf{F}_j + \frac{c}{|\Omega_j|} \sum_k l_{jk} E_{jk} \mathbf{n}_{jk} = \frac{c\sigma_c}{|\Omega_j|} \int_{\Omega_j} \mathbf{F} \end{cases}$$

Dans lequel les flux numériques  $(\mathbf{F}_{jk}, \mathbf{n}_{jk})$  et  $E_{jk}\mathbf{n}_{jk}$  entre une maille  $j$  et sa voisine  $k$  sont définis comme suit<sup>9</sup> :

$$\begin{aligned} (\mathbf{F}_{jk}, \mathbf{n}_{jk}) &= l_{jk} M_{jk} \left( \frac{\mathbf{F}_j^n \cdot \mathbf{n}_{jk} + \mathbf{F}_k^n \cdot \mathbf{n}_{jk}}{2} - \frac{E_k^n - E_j^n}{2} \right) \\ E_{jk} \mathbf{n}_{jk} &= l_{jk} M_{jk} \left( \frac{E_j^n + E_k^n}{2} - \frac{\mathbf{F}_k^n \cdot \mathbf{n}_{jk} - \mathbf{F}_j^n \cdot \mathbf{n}_{jk}}{2} \right) \mathbf{n}_{jk} \end{aligned}$$

Avec :

$$\begin{aligned} M_{jk} &= \frac{2}{2 + \Delta x \sigma_{jk}} \\ \sigma_{jk} &= \frac{1}{2} \left( \sigma_c(\rho_j, T_j^n) + \sigma_c(\rho_k, T_k^n) \right) \end{aligned}$$

9. Ces définitions sont adaptées de (FRANCK, 2012, p. 161)

À l'itération  $n$ , l'étape 2 du schéma de « splitting » s'écrit donc sous la forme :

$$\begin{cases} \frac{E_j^{n+1} - E_j^*}{\Delta t} + \frac{c}{|\Omega_j|} \sum_k (\mathbf{F}_{jk}, \mathbf{n}_{jk}) = 0 \\ \frac{\mathbf{F}_j^{n+1} - \mathbf{F}_j^*}{\Delta t} + \frac{c}{|\Omega_j|} \sum_k E_{jk} \mathbf{n}_{jk} - c \mathbf{S}'_j = c \mathbf{S}_j \end{cases}$$

Avec :

$$\begin{aligned} \mathbf{S}_j &= - \left( \sum_k M_{jk} \sigma_{jk} \right) \mathbf{F}_j^{n+1} \\ \mathbf{S}'_j &= \frac{1}{|\Omega_j|} \left( \sum_k l_{jk} M_{jk} \mathbf{n}_{jk} \right) E_j^n \end{aligned}$$

Qui se réécrit comme suit :

$$\begin{cases} E_j^{n+1} = E_j^* + \alpha \sum_k (\mathbf{F}_{jk}, \mathbf{n}_{jk}) \\ \mathbf{F}_j^{n+1} = \beta \mathbf{F}_j^* + \gamma E_j^n + \delta \sum_k E_{jk} \mathbf{n}_{jk} \end{cases} \quad (3.8)$$

Avec :

$$\begin{aligned} \alpha &= -\frac{c\Delta t}{|\Omega_j|}, \quad \beta = \frac{1}{\Delta t} \left( \frac{1}{\Delta t} + c \sum_k M_{jk} \sigma_{jk} \right)^{-1}, \quad \gamma = \frac{c}{|\Omega_j|} \left( \frac{1}{\Delta t} + c \sum_k M_{jk} \sigma_{jk} \right)^{-1} \left( \sum_k l_{jk} M_{jk} \mathbf{n}_{jk} \right) \\ \text{et } \delta &= -\frac{c}{|\Omega_j|} \left( \frac{1}{\Delta t} + c \sum_k M_{jk} \sigma_{jk} \right)^{-1} \end{aligned}$$

La condition de CFL  $\Delta t < \frac{\Delta x}{c}$  est nécessaire pour assurer la stabilité du schéma. Lors de l'implémentation en C++, on remarquera qu'en pratique, il faut prendre  $\Delta t < 0.5 \times \frac{\Delta x}{c}$ .

### 3.3 Implémentation en C++

Le code de calcul 2D a été développé durant la cinquième semaine du stage. En fournissant des paramètres (physiques, géométriques, etc.) au programme, l'on peut récupérer les signaux temporels  $E$ ,  $F$  et  $T$  sur les quatre bords du domaine. Il permet aussi d'exporter les signaux sur l'entièreté du domaine en tout temps. Ces derniers signaux peuvent ensuite être visualisés sous forme d'animation à l'aide d'un notebook Jupyter construit à cet effet. L'exécutable pour effectuer des simulations est disponible avec le reste du code sur le dépôt Github [projet-inverse-2d](#).

#### 3.3.1 Configuration de la simulation

L'exécutable nécessite un fichier de configuration pour s'exécuter (.cfg, .txt, etc.). Les paramètres obligatoires sont indiqués à la figure 3.3. Les unités sont celles de la liste des symboles de la page 1. Des détails supplémentaires sont donnés en annexe A.

```

x_min 0
x_max 1
y_min 0
y_max 1
N 90

c 299
a 0.01372
C_v 0.14361

CFL 0.5
precision 1e-6
t_0 0
t_f 0.01

rho crenau(0.5,0.5,0.1,10)
sigma_a rho*T
sigma_c rho*T

E_0 0.01372*(5^4)
F_0_x 0
F_0_y 0
T_0 5

E_u neumann
F_u_x neumann
F_u_y neumann
T_u neumann

E_d neumann
F_d_x neumann
F_d_y neumann
T_d neumann

E_l ponctuel(0.4,0.6)
F_l_x 0
F_l_y 0
T_l 5

E_r neumann
F_r_x neumann
F_r_y neumann
T_r neumann

export_file data/df_simu.csv
export_mode dataframe
write_mode truncate

```

FIGURE 3.3 – Exemple d’un fichier de configuration. Ici ne sont représentés que les 38 paramètres obligatoires pour faire tourner une simulation et l’exporter. On remarque que le nombre de mailles en horizontale  $M$  n’est pas inclu ; il est calculé automatique afin d’obtenir un maillage uniforme. Le résultat produit par ce fichier est présenté aux figures 3.4 et 3.5.

### 3.3.2 Sauvegarde des données

Comme mentionné ci-haut, on dispose de deux options pour sauvegarder les résultats de la simulation :

- Sous le forme **CSV** : ce mode permet une visualisation facile des résultats à l’aide de notebook. Il est très couteux en espace mémoire et nécessite la librairie Pandas pour une lecture sous forme de dataframe. Cette opération est lente et prend une quantité non négligeable de RAM, ce qui peut nuire à l’usage qu’on veut faire des données.
- Sous le format **SDS**<sup>10</sup> : ce format binaire ne sauvegarde que les informations les plus importantes de la simulation. En l’occurrence la source utilisée, la densité du domaine, et les différents signaux sur les bords du domaine. Il est particulièrement intéressant pour générer les données nécessaires à l’apprentissage. Les détails concernant ce format sont donnés en annexe A.

## 3.4 Résultats

Quelques résultats obtenus sont présentés ici. Le premier résultat (figures 3.4 et 3.5) est obtenu avec le fichier de configuration de la figure 3.3. La source est une onde sinusoïdale placée en  $E$  sur une portion de la gauche. La densité a la forme d’un signal en créneau égale à 10 sur le créneau et 0.1 en dehors. Les opacités d’absorption et de dispersion sont proportionnelles à la densité.

10. Source - Densité - Signal



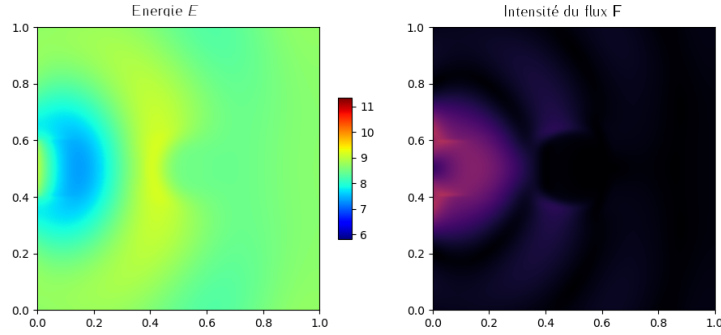


FIGURE 3.4 – Visualisation de l'énergie et de l'intensité du flux des photons au temps final pour un domaine avec une densité en forme de crêneau circulaire (vu du haut). Cette figure correspond au résultat obtenu avec la configuration 3.3.

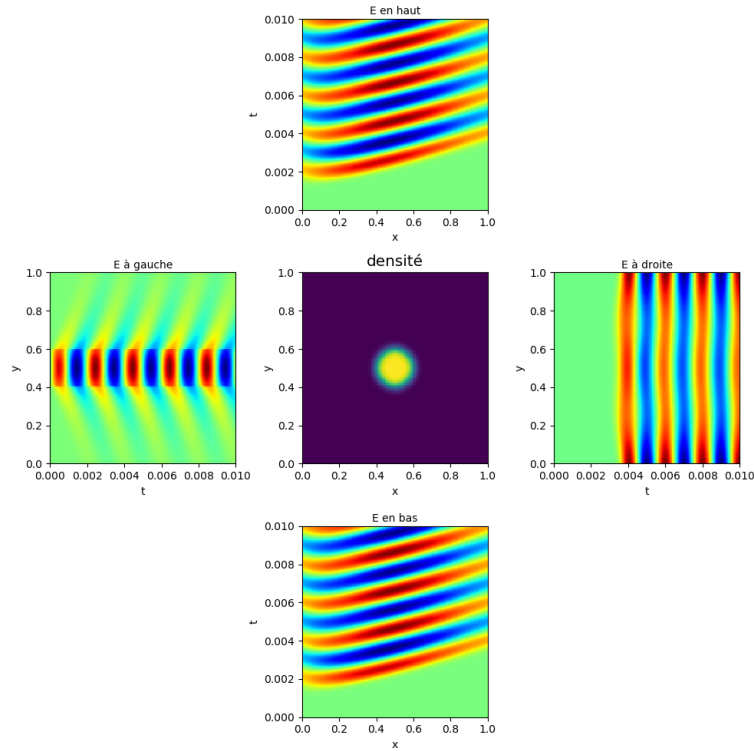


FIGURE 3.5 – Évolution de l'énergie sur les bords (vue de haut). La densité est illustrée au milieu de l'image, et l'évolution de l'énergie  $E$  est présentée aux alentours (correspondant au signal sur chacun des quatre bords du bas, du haut, de gauche et de droite mesuré au cours du temps). Les indices  $x$ ,  $y$ , et  $t$  représentent respectivement l'abscisse, l'ordonnée et le temps. Les autres figures associées à ce cas sont 3.3 et 3.4.

La figure 3.4 permet d'observer une absorption presque totale du signal au niveau du crêneau due à la forte valeur des opacités d'absorption et de dispersion. En ce sens, le crêneau (saut de densité) très opaque agit comme un obstacle à la propagation du signal. L'évolution de l'énergie sur les bords du domaine (figure 3.5) est tout aussi importante, mais traduit moins bien cet effet<sup>11</sup> qu'à la densité sur la propagation du signal.

11. Très peu perceptible pour l'œil humain, mais probablement pas pour un réseau de neurones.

Nous testons ensuite notre modèle sur le cas très particulier de l'approximation de diffusion (figures 3.6, 3.7 et 3.8). Le bord gauche est continûment chauffé ( $E_{gauche} = a(T_{initial} + 1)^4$ ). La densité est un signal en forme de créneau rectangulaire (vu du haut) valant 10. En dehors de ce créneau, la densité vaut 0.1. Théoriquement, la limite de diffusion s'observe pour  $c/\sigma_c \approx 1$  (voir équation 3.3), mais cela pose des problèmes de visualisation en utilisant les coefficients que nous avons choisis. On prend donc  $\sigma_a = \sigma_c = 100 \times \rho$  ce qui donne  $c/\sigma_c \approx 30$  en dehors de l'obstacle et  $c/\sigma_c \approx 0.3$  sur l'obstacle.

```
x_min 0
x_max 1
y_min 0
y_max 1
N 90

c 299
a 0.01372
C_v 0.14361

CFL 0.5
precision 1e-6
t_0 0
t_f 0.01

rho crenau(0.5,0.5,0.1,10)
sigma_a rho*100
sigma_c rho*100

E_0 0.01372*(5^4)
F_0_x 0
F_0_y 0
T_0 5

E_u neumann
F_u_x neumann
F_u_y neumann
T_u neumann

E_d neumann
F_d_x neumann
F_d_y neumann
T_d neumann

E_l 0.01372*(6^4)
F_l_x 0
F_l_y 0
T_l 5

E_r neumann
F_r_x neumann
F_r_y neumann
T_r neumann

export_file data/df_simu.csv
export_mode dataframe
write_mode truncate
```

FIGURE 3.6 – Configuration utilisée pour illustrer la limite de diffusion. L'option pour obtenir un obstacle en forme de rectangle n'est pas insérable dans le fichier de configuration, cela se fait directement dans le code de calcul.

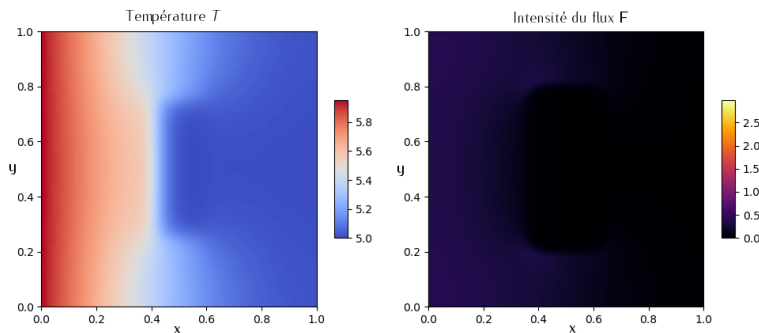


FIGURE 3.7 – Visualisation de la température du domaine et de l'intensité du flux des photons au temps final pour la limite de diffusion. La densité à une forme de créneau rectangulaire comme représenté sur l'image du milieu de la figure 3.8.

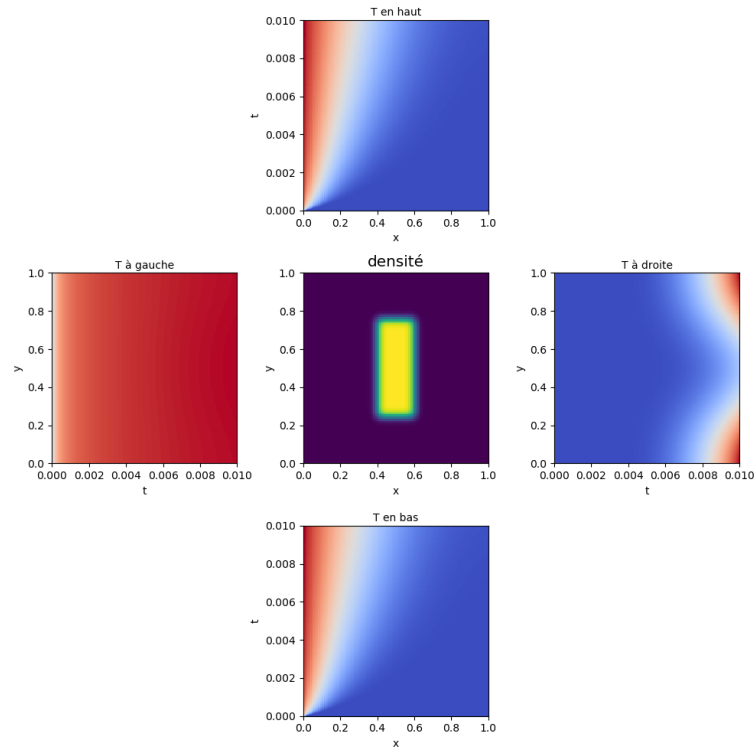


FIGURE 3.8 – Évolution de la température sur les bords illustrant l’effet de diffusion. Tout comme à la figure 3.7, l’expression des opacités pour cette simulation est  $\sigma_a = \sigma_c = 100 \times \rho$  afin d’obtenir un maximum de diffusion en dehors de l’obstacle mais une absorption totale sur l’obstacle.

On confirme effectivement l’effet de diffusion du signal dans le domaine. Nous pouvons à présent passer à l’apprentissage. Nous nous placerons dans des conditions proches de celles de la configuration 3.3. L’apprentissage utilisera aussi les données de simulation 1D non présentées ici. Les résultats 1D ont été présentés et analysés dans le rapport de fin de projet de CSMI.

## Chapitre 4

# Apprentissage

L'objectif de cette section est de reconstruire la densité de la matière en connaissant l'énergie  $E$ , le flux  $F$ , et la température  $T$  sur les bords du domaine au cours du temps. Il s'agit donc d'un problème inverse de régression<sup>1</sup>, la densité faisant partie des paramètres du problème direct; et les signaux temporels  $E$ ,  $F$ , et  $T$  faisant partie des résultats. Nous ferons une simplification de taille : **la densité est un signal en créneau lissé**<sup>2</sup> (on utilisera un créneau cylindrique i.e ayant la forme d'un cercle de diamètre 0.1 vu du haut). Ainsi, reconstruire la densité revient juste à prédire la position et la hauteur du créneau. La valeur de la densité en dehors du créneau sera connue.

Nous recherchons donc une fonction  $f^{-1}$  (inverse de la fonction  $f$  définissant le problème direct) telle que  $y = f^{-1}(\mathbf{X})$ . Où  $y$  représente la densité de la matière (plus précisément les attributs de son créneau), et  $\mathbf{X}$  les signaux sur les bords. Le caractère généralement mal posé des problèmes inverses rend difficile la détermination de  $f^{-1}$ . On procède donc à une approximation de  $f^{-1}$  notée  $\hat{f}^{-1}$ , par apprentissage supervisé<sup>3</sup> à l'aide d'un ANN<sup>4</sup>. En notant  $\theta$  les paramètres de l'ANN, on cherche  $\hat{y}$  telle que  $\hat{y} = \hat{f}^{-1}(\mathbf{X}, \theta)$ .

### 4.1 Description des entrées/sorties

Il s'agit dans cette partie de décrire les entrées et les sorties qui ont été obtenues durant l'étape (préliminaire à apprentissage) de génération des données. En 1D le domaine est  $[0, 1]$  discrétisé sur 300 mailles, et en 2D le domaine est  $[0, 1] \times [0, 1]$  discrétisé sur 28x28 mailles. Cependant les données 1D et 2D se partagent un certain nombre de paramètres, principalement composé des paramètres physiques du problème :

- temps final = 0.01 sh (1 *shake* (sh) =  $10^{-8}$  secondes)
- $c = 299$  [cm/sh]
- $a = 0.01372$  [g/cm/sh<sup>2</sup>/keV]
- $C_v = 0.14361$  [Jerk/g/keV] (1 Jerk = 1m/s<sup>3</sup>)
- la densité  $\rho$  est un signal en créneau [g cm<sup>-3</sup>]
- $\sigma_a = \rho T$  [cm<sup>-1</sup>]
- $\sigma_c = \rho T$  [cm<sup>-1</sup>]
- $T_0 = 5$  [keV] (en termes de température, 1 keV = 11605 K)
- $E_0 = 0.01372 \times 5^4$  [g/cm/sh<sup>2</sup>]
- $\mathbf{F}_0 = \mathbf{0}$  [g/sh<sup>2</sup>]

La situation initiale pour chacune des simulations effectuées correspond à l'équilibre radiatif ( $E_0 = aT_0^4$ , i.e température initiale de la matière ( $T_0$ ) = température initiale de la radiation ( $Tr_0$ ) =

1. Une classification peut tout aussi bien être implémentée (voir paragraphe 4.5.2). Sauf mention du contraire, lorsqu'on parlera d'apprentissage, il s'agira du problème de régression.

2. Un créneau sur la densité sera aussi appelé saut de densité, ou pic de densité, ou obstacle sur la densité

3. Un apprentissage automatique pour lequel les exemples (ou « samples ») sont annotés par des labels.

4. Réseau de neurones artificiel

$(E_0/a)^{1/4}$ ). La situation sur une portion de la gauche correspond à cet équilibre perturbé par une source sinusoïdale ( $E_{\text{gauche}} = aT_0^4 + 5 \sin(2k\pi t)$ )<sup>5</sup> (condition de Dirichlet). Sur le reste de la gauche (en 2D), l'équilibre radiatif est maintenu (condition de Dirichlet). Les situations sur les autres bords (1 en 1D et 3 en 2D) correspondent à des sorties libres (condition de Neumann).

À quelques différences près (nombre de mailles, position de la source sur la gauche), la figure 3.3 représente un exemple typique de configuration utilisé en 2D (les paramètres physiques applicables en 1D peuvent aussi être observés). La définition des fichiers de configurations pour la 1D et la 2D est définie sur leur dépôt Github respectifs (voir annexe A).

#### 4.1.1 En 1D

L'aspect d'une entrée et d'une sortie 1D est représentée à la figure 4.1. Les entrées sont composées des signaux temporels  $E$ ,  $F$ , et  $T$ . Suivant chacun de ces canaux, il faut normaliser les entrées avant de les nourrir au réseau de neurones. Vu que le signal provient de la gauche, les entrées ne sont constituées que du signal récupéré sur le bord droit du domaine. La taille d'un exemple est présentée à la figure 4.2.

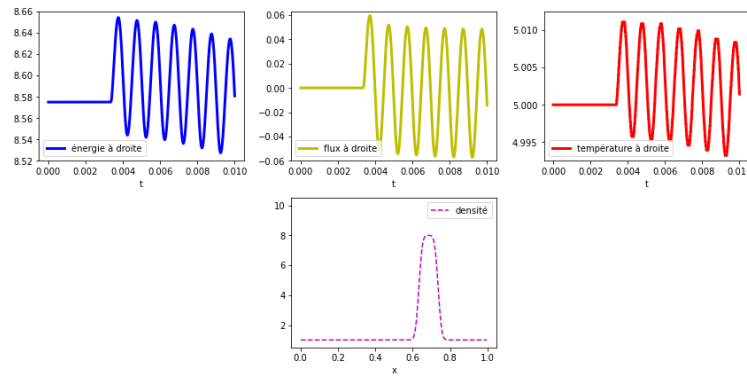


FIGURE 4.1 – Visualisation d'une entrée de l'ANN (en haut) et d'une sortie (en bas) en 1D. Seul le signal sur la droite du domaine est utilisé. Tout les 3 canaux  $E$ ,  $F$  et  $T$  sont représentés ici.

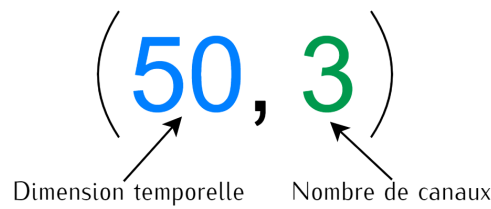


FIGURE 4.2 – Taille d'une entrée 1D. La dimension temporelle (nombre d'itérations) a été ré-échantillonnée de 907 à 50. Les 3 canaux désignent les signaux  $E$ ,  $F$  et  $T$ .

Comme mentionné plus haut, nous avons fait quelques simplifications sur la nature de la sortie. Il s'agit d'un vecteur de seulement deux scalaires<sup>6</sup> représentant la position et la hauteur du saut de densité :

- **Position** : il s'agit de l'abscisse de l'obstacle choisi de façon à ce que le créneau se situe entièrement dans le domaine (i.e position de son centre comprise dans l'intervalle  $[0.07, 0.92]$ ).

5. Toute la gauche est naturellement couverte par la source en 1D, avec  $k = 1000$ . En 2D,  $k = 500$ .

6. Les deux scalaires sont extraits de l'image du bas de la figure 4.1

- **Hauteur** : comprise dans l'intervalle  $[1.05, 9.99]$ . La valeur de la densité en dehors du créneau est de 1.

### 4.1.2 En 2D

Une entrée 2D contient considérablement plus d'information qu'une entrée 1D. Les 3 signaux  $E$ ,  $F$  et  $T$  sur les 4 bords sont inclus. On y ajoute chacun des groupes de signaux correspondants aux 4 positions de la source sur la gauche. En effet, une entrée correspond à 4 simulations effectuées chacune avec la source à une position différente, comme on peut le voir à la figure 4.3. La taille d'une entrée est donnée à la figure 4.4.

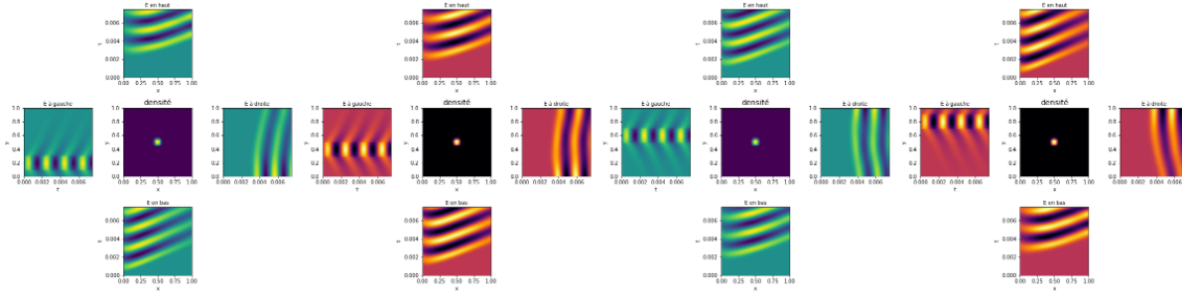


FIGURE 4.3 – Visualisation d'une entrée (aux alentours de chacune des quatre images) et d'une sortie (aux milieux) en 2D. On peut voir les positions des 4 sources (sur la gauche) utilisées à tour de rôle pour former une seule entrée. Ici n'est représentée que l'énergie (qui constitue 1/3 des canaux) sur les 4 bords. Les images des densités ci-contre ont été obtenues par interpolation bilinéaire d'une image initiale très grossière (28x28).

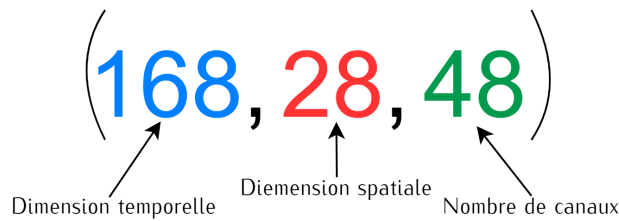


FIGURE 4.4 – Taille d'une entrée en 2D. Contrairement à la 1D, il faut tenir compte de la dimension spatiale qui correspond au nombre de mailles sur chaque bord du domaine rectangulaire ( $N = M = 28$ ). Le nombre de canaux passe à 48 car il faut déjà tenir compte des 3 canaux originaux ( $E$ ,  $F$ , et  $T$ ), ensuite de chacun de 4 bords du domaine, et enfin des 4 positions de la source.

Compare à la 1D, il faut rajouter l'ordonnée du saut de densité à la liste des scalaires prédits. Une sortie donc est un vecteur de taille 3. Afin d'éviter des cas trop extrêmes, les valeurs observées (ou labels pour l'apprentissage) sont convenablement choisies au moment de la génération des données :

- **Abscisse** : comprise dans l'intervalle  $[0.2, 0.8]$  afin de récupérer une réponse non aberrante sur chacun des 4 bords du domaine.
- **Ordonnée** : comprise aussi dans l'intervalle  $[0.2, 0.8]$ .
- **Hauteur** : comprise dans l'intervalle  $[0.1, 10]$ . La valeur 0.1 est aussi la valeur de la densité en dehors de son créneau. Cette valeur faible permet d'éviter l'absorption complète de l'onde sur le domaine.

En 1D comme en 2D, il faudra séparer le jeu de données en 3 parties avant de procéder à l'apprentissage :

- **train** : il s'agit des données utilisées pour l'entraînement du modèle d'ANN construit.
- **val** : pour évaluer la qualité d'une époque de l'entraînement.
- **test** : pour tester le modèle. Les scores que nous présenterons seront calculés sur ces données.

Des jeux de données complets 1D/2D ont été sauvegardés sur Github ; et les détails pour les récupérer et les traiter sont donnés en annexe B.

## 4.2 Architecture générale

Un réseau de neurones artificiel<sup>7</sup> (ANN) est un système computationnel basé sur le réseau de neurones biologique. L'apprentissage profond, à travers ses réseaux de neurones profonds<sup>8</sup> (DNN<sup>9</sup>), permet de résoudre des problèmes en Machine Learning que les méthodes classiques (régression linéaire, SVM, etc.) ne peuvent pas. Il réussit cela en introduisant des représentations des données qui s'expriment sous forme d'autres représentations, plus simples cette fois.

Les réseaux profonds en aval<sup>10</sup> (ou MLP<sup>11</sup>) constituent l'exemple de choix en apprentissage profond. On peut l'interpréter comme une fonction (plus exactement une composition de fonctions :  $f^{-1} = f_1 \circ f_2 \circ \dots \circ f_n$ ) faisant correspondre une série d'entrées à une série de sorties. Il est constitué de plusieurs couches (assimilables aux fonctions  $f_1, f_2, \dots, f_n$  précédentes) apprenant chacune un aspect particulier des données. On distingue une couche d'entrée, une ou plusieurs couches cachées, et une couche de sortie (voir figure 4.5).

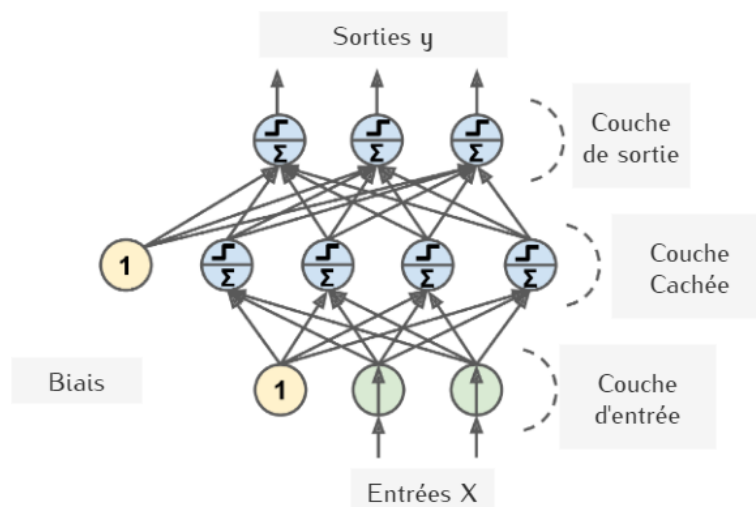


FIGURE 4.5 – Illustration d'un MLP avec une couche cachée. Le nombre de couches cachées peut être élevée ce qui conduit aux DNN. Le 1 représente le biais (GERON, 2017, p. 286).

Les réseaux de neurones convolutifs (CNN) sont une forme de MLP spécialisés dans le traitement des données en forme de grille. Par exemple des séries en temps qui peuvent être vues comme des grilles 1D (l'axe de temps) prenant des données (vecteur de données) à intervalle de

7. Ou juste réseau de neurones dorénavant

8. Réseaux de neurones composé d'un nombre relativement élevé de couches cachées

9. Deep Neural Network

10. En opposition à un réseau de neurones récurrent qui réutilise les résultats de son modèle pour s'améliorer.

11. Multi-Layer Perceptron

temps régulier (GOODFELLOW et al., 2016). Ils sont donc particulièrement adaptés à la reconstruction de la densité partant des signaux temporels  $E$ ,  $F$ , et  $T$ .

L'architecture de CNN de base pour notre apprentissage a été proposée par M. VIGON. Nous utiliserons deux variantes : DRNN<sup>12</sup> 1 (figure 4.6) et DRNN 2 (figure 4.7). Les architectures seront implémentées sous la librairie de Machine Learning Keras (avec le « backend » TensorFlow). Les différentes couches présentées seront détaillées dans la suite. Nous indiquerons aussi en quoi elles sont importantes pour notre apprentissage.

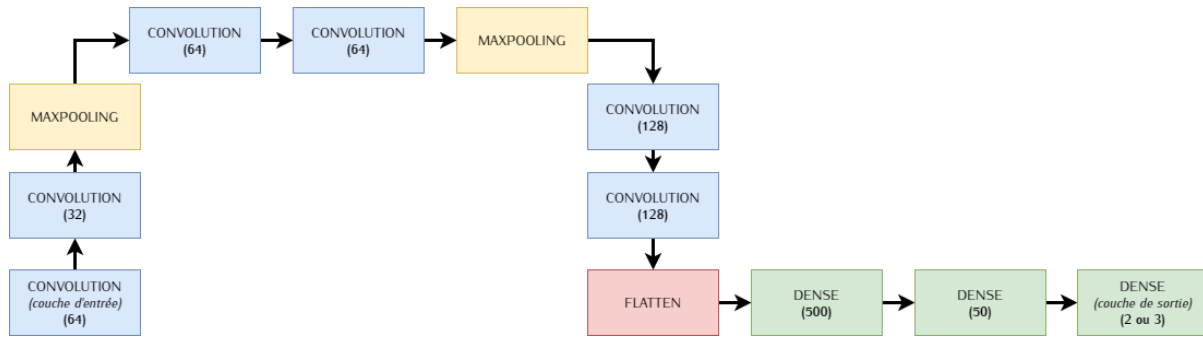


FIGURE 4.6 – Première architecture (nommée DRNN 1). Le nombre de neurones utilisés pour chaque couche est indiqué entre parenthèses. Le nombre de neurones de la couche de sortie dépend qu'on soit en 1D ou en 2D.

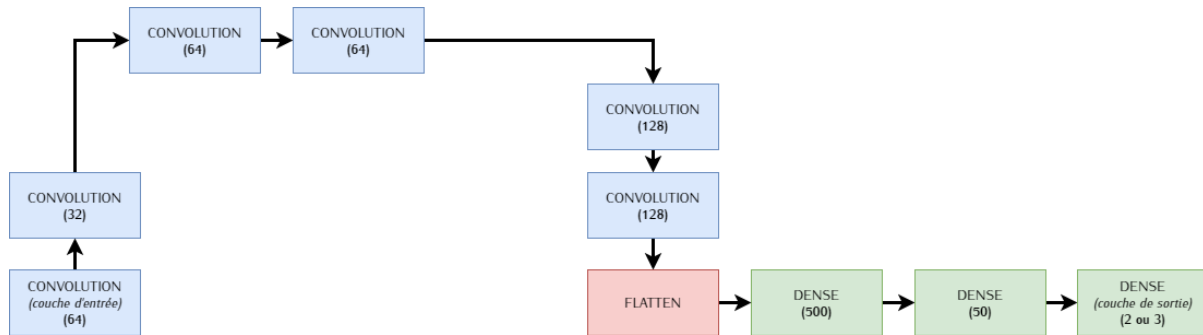


FIGURE 4.7 – Deuxième architecture utilisée (nommée DRNN 2). Ce modèle ne contient pas de couche de « pooling »<sup>13</sup>.

## 4.3 Les couches utilisées

### 4.3.1 Les couches de convolution

La convolution est l'opération fondamentale d'un CNN. Il s'agit d'une opération linéaire qui combine deux signaux pour en extraire un troisième. En général, une opération de convolution se définit par la formule suivante ( $i$  est le signal d'entrée et  $k$  est le noyau de la convolution) :

$$s(t) = (i * k)(t) = \int i(x)k(t - x) dx$$

12. Density Reconstruction Neural Network

6. Les détails concernant l'opération de « pooling » seront donnés à la section 4.3.2



En pratique, les signaux temporels ne sont pas continus, ils sont discrétisés par intervalles de temps  $\Delta t$ . Dans ce contexte, la convolution 1D se définit par la formule :

$$s(t) = \sum_{x=-\infty}^{\infty} i(x)k(t-x) \quad (4.1)$$

Cette formule doit être adaptée en 2D vu qu'on fera aussi un apprentissage 2D. La formule devient donc :

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m,j-n) \quad (4.2)$$

L'opération de convolution est commutative grâce à l'inversion du noyaux relativement par rapport au signal d'entrée. Cette propriété, bien qu'importante d'un point de vue théorique, ne présente pas d'avantages du point de vue computationnel. C'est la raison pour laquelle on dispose de l'opération de cross-corrélation qui est une convolution sans inversion du noyau. En 2D elle se présente comme ceci :

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m,j+n)K(m,n) \quad (4.3)$$

On remarque aussi que le parcours des indices se fait suivant l'input. Il se trouve que c'est plus direct et rapide ainsi, parce qu'il y a moins de variation dans la plage de valeurs valides pour  $m$  et  $n$  (GOODFELLOW et al., 2016, 322ff.).

Plusieurs bibliothèques de Machine Learning implémentent la cross-corrélation mais l'appellent convolution. C'est le cas de Keras lorsqu'elle utilise le « backend » TensorFlow<sup>14</sup> (KERAS-TEAM, 2017).

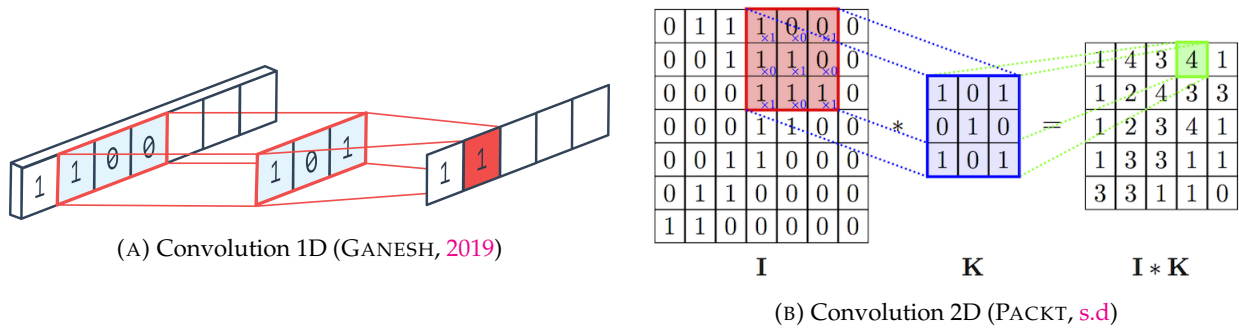


FIGURE 4.8 – Illustration d'une convolution (cross-corrélation) 1D/2D en mode "valid" (aucun « padding » de 0 ne sera ajouté et la sortie  $S$  aura une taille inférieure à l'entrée  $I$ ). La taille du noyaux que nous utiliserons sera de 3 (en 1D) et (6,2) (en 2D). Un autre paramètre important pour réduire la taille de la sortie est le "stride"; il caractérise de l'écart entre deux applications consécutives du noyaux de convolution  $K$ . Nous le prenons égale à 1 (en 1D) et (1,1) (en 2D) de façon à couvrir tous les indices valides de l'entrée  $I$ .

Dans les architecture de CNN typiques, l'opération de convolution est généralement suivi d'une étape dite de détection. Dans cette étape, les résultats de la convolution (linéaires) sont passés à une fonction non linéaire au niveau d'une couche d'activation. Nous détaillerons la notion d'activation dans les sections suivantes. Après cette étape de détection, le Pooling est généralement appliqué pour modifier les résultats encore plus profondément.

14. Lorsqu'on utilise Theano, les convolutions sont effectivement des convolution comme définie en 4.1 et 4.2.

### 4.3.2 Le Max-pooling

L'opération de Pooling permet de réduire la taille des données (« downsampling »). Une fonction de Pooling transforme les entrées voisines par une fonction d'agrégation statistique. Plusieurs fonctions d'agrégations peuvent être utilisées. Par exemple, le Max-pooling renvoie le maximum sur un domaine appelé « pool size » (rectiligne en 1D et rectangulaire en 2D).

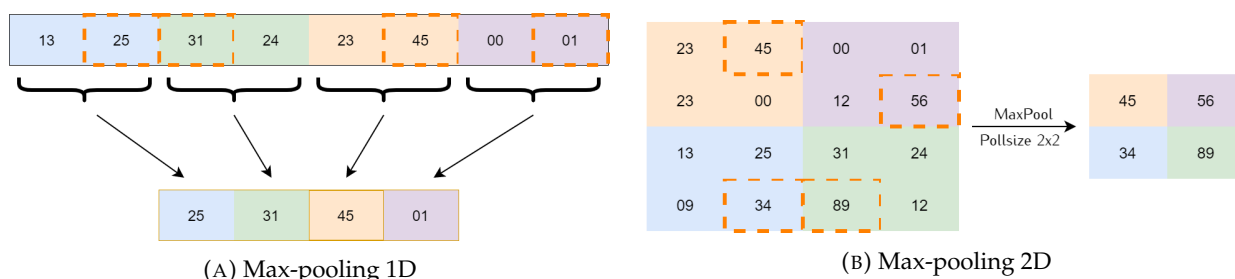


FIGURE 4.9 – Opération de Max-pooling en 1D/2D avec un « pool size » de 2 (en 1D) et de 2x2 (en 2D).

En général, l'opération de Pooling permet de rendre la représentation approximativement invariante aux petites variations dans l'input. Parlant de l'identification d'objets dans une image par exemple, « L'invariance par translations locales (petites translations) peut être utile si on est plus intéressé par la présence de l'objet que par sa localisation exacte. » (GOODFELLOW et al., 2016, p. 330).

Dans le problème inverse que nous résolvons, on aimerait non seulement détecter la présence du saut de densité, mais aussi ses coordonnées exactes. Cela nous amène donc à considérer dans un premier temps une architecture avec Max-pooling (figure 4.6), et dans un deuxième temps, sans Max-pooling (figure 4.7).

### 4.3.3 Flatten

L'opération d'aplatissage (ou Flatten) permet de transformer les données en quittant de la forme tensorielle (2D avec plusieurs canaux) à une forme vectorielle. Il s'agit en réalité d'une étape de préparation pour les couches denses (complètement connectées).

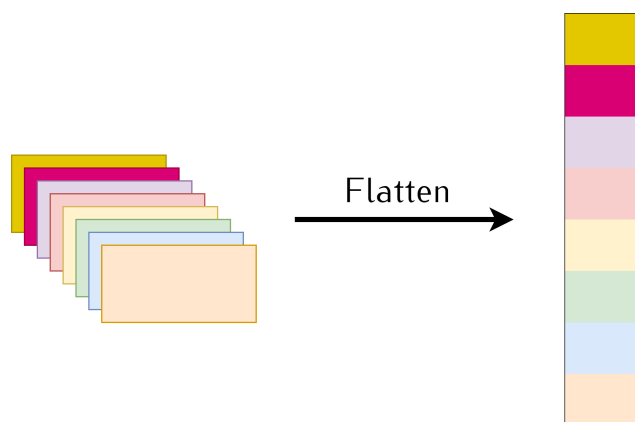


FIGURE 4.10 – Illustration d'une opération de Flatten). Qu'on soit en 1D ou en 2D, le Flatten assure que sa sortie est un vecteur sur lequel une couche dense peut opérer.

### 4.3.4 Les couches denses

À ce niveau, tous les neurones d'une couche sont connectés à tous les neurones de la couche précédente. Une couche dense (ou « fully connected ») prend les résultats d'une convolution/Pooling et en ressort des poids. Les couches de convolution ayant appris des aspects particuliers des données, la couche dense est un moyen relativement facile d'apprendre des combinaisons de ces dernières.

Si  $f_i$  désigne la fonction représentant une couche dense, l'opération effectuée est  $f_i(\mathbf{X}) = \phi(\mathbf{XW} + \mathbf{b})$ , où  $\mathbf{X}$  représente les entrées de la couche,  $\mathbf{b}$  le biais,  $\mathbf{W}$  la matrice des poids (une ligne par neurone d'entrée et une colonne par neurone de cette couche, excepté ceux du biais), et  $\phi$  la fonction d'activation que nous détaillerons plus tard. (GERON, 2017, p. 286)

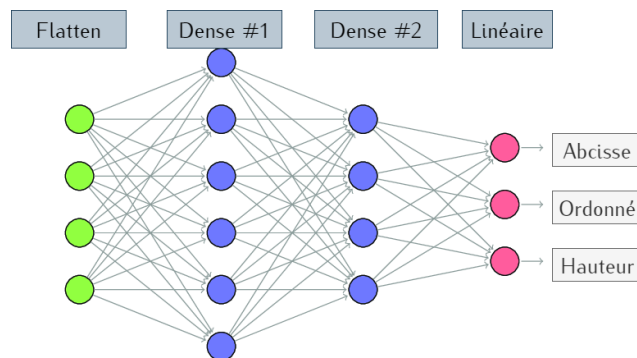


FIGURE 4.11 – Série de 3 couches denses « fully connected ». La couche en vert représente le résultat de l'opération de Flatten susmentionnée. L'utilisation des couches denses constitue la dernière étape des réseaux en figures 4.6 et 4.7 (Image adaptée de (FAUSKE, 2006))

## 4.4 Configuration de l'entraînement

Keras propose une multitude d'options et d'hyper-paramètres pour tuner le modèle et son entraînement. Ceux qui ont été utilisés sont indiqués dans les sections suivantes.

### 4.4.1 Les métriques

#### 4.4.1.1 Loss MSE

Pendant la génération des données, on a pris soin de ne pas introduire de données aberrantes. La MSE (Mean Squared Error) qui est plus élevée que la MAE (Mean Absolute Error) sur les valeurs aberrantes est donc la plus adaptée. Si les  $\hat{y}_i$  désignent les prédictions et  $y_i$  les véritables cibles (ou valeurs observées ou labels), la MSE se définit par :

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (4.4)$$

#### 4.4.1.2 Coefficient de détermination $R^2$

Le coefficient de détermination  $R^2$  est très important en statistique. On peut l'obtenir par la formule :

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \quad (4.5)$$

Avec

$$SS_{res} = \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad \text{et} \quad SS_{tot} = \sum_{i=1}^n (y_i - \bar{y})^2$$

Où  $\bar{y} = \sum_{i=1}^n y_i$  représente la moyenne des valeurs observées.

On peut remarquer que :

- Si le modèle prédit les valeurs attendues (observées), le score  $R^2$  vaut 1.
- Si le modèle prédit toujours la valeur moyenne  $\bar{y}$ , le score  $R^2$  vaut 0.
- Si les prédictions sont pires que la moyenne, le score  $R^2$  est négatif.

En général, si les prédictions et les valeurs observées sont très corrélées (sans être égales), on aura un score  $R^2$  qui n'est pas caractéristique des résultats. En effet, pour des tâches de régression, il se définit comme étant le carré du coefficient de corrélation entre les valeurs prédites et les valeurs observées.

Dans la suite de ce rapport, le score  $R^2$  sera présenté sous forme de pourcentage.

#### 4.4.1.3 Un score personnalisé

On définit à présent un score particulièrement adapté à nos données. On déclare qu'une prédiction est correcte si elle est suffisamment proche du label :

- **au dixième près** pour la position (suivant  $x$  ou  $y$ ) car le domaine d'étude est  $[0, 1]$  en 1D et  $[0, 1] \times [0, 1]$  en 2D
- **à l'unité près** pour la hauteur car les hauteurs observées sont comprises entre 1 et 10 (en 1D) et 0.1 et 10 (en 2D)

Le score personnalisé est donc un score sévère (pourcentage des prédictions correctes) qui récompense les prédictions qui sont à la fois précises en hauteurs et en position.

### 4.4.2 Les hyper-paramètres

#### 4.4.2.1 L'optimiseur

L'optimisation est une méthode d'accélération de l'entraînement. À la place de l'algorithme classique de descente de gradient stochastique, nous utiliserons l'optimiseur Adam<sup>15</sup> qui combine les propriétés de deux autres algorithmes d'entraînement (AdaGrad et RMSProp) pour offrir une performance de taille.

#### 4.4.2.2 Activation ReLU

La fonction d'activation introduit une non-linéarité entre les couches. L'avantage majeur de l'activation ReLU<sup>16</sup> (que nous utiliserons) par rapport aux autres fonctions d'activation c'est qu'elle n'active pas tous les neurones en même temps. D'un point de vue computationnel, elle est très efficace tout en produisant des résultats satisfaisants.

15. Adaptive moment estimation

16. Rectified Linear Unit

#### 4.4.2.3 Le taux d'apprentissage

Il s'agit du paramètre le plus influant pour notre apprentissage. Il contrôle à quelle vitesse le modèle s'adapte au problème, en déterminant de quelle quantité les poids des neurones seront mis à jour<sup>17</sup> après l'algorithme de « backpropagation ». S'il est très élevé, il peut rapidement conduire à solution non optimale. S'il est très faible, le modèle peut resté figé (il faudra alors un nombre élevé d'époques pour potentiellement le débloquent). En 1D, nous avons utilisé un taux d'apprentissage égale à 1e-4 alors qu'en 2D, nous l'avons pris égale à 1e-5.

#### 4.4.2.4 Le « batch size »

Il s'agit de la taille de chaque paquet de données<sup>18</sup> passé au modèle durant une époque. Un « batch size » faible apporte du bruit au modèle, dû au fait qu'une portion aléatoire des données est utilisée pour mettre à jour les poids des neurones. Ceci assure une meilleure généralisation du modèle tout en limitant la quantité de données chargée dans la RAM à chaque époque.

#### 4.4.2.5 Early-stopping

La technique d'Early-stopping sera notre moyen primaire de lutte contre le sur-apprentissage<sup>19</sup>. Pour l'implémenter de façon efficace sous Keras, il nous faut une variable appelée "patience". Il s'agit du nombre d'époques à attendre avant d'arrêter l'apprentissage de façon précoce. Nous arrêterons nos entraînements dès que le score  $R^2$  (voir paragraphe 4.4.1.2) sur le jeu de validation (val) n'aura pas augmenté pendant 10 époques.

### 4.5 Résultats

Nous résumons la section précédente en spécifiant les hyper-paramètres (et leurs définitions) utilisés pour entraîner le modèle sous Keras.

TABLE 4.1 – Liste des paramètres majeurs utilisés pour l'entraînement. L'activation "relu" est utilisée sur les couches cachées et "linear" sur la couche de sortie. Les paramètres non indiqués dans ce tableau (« pool size », etc.) sont pris par défaut sous Keras.

Hyper-paramètre	Définition	Valeur 1D / 2D
learning_rate	taux d'apprentissage	1e-4 / 1e-5
batch_size	taille d'un batch a chaque époque	32
optimizer	algorithme d'optimisation	Adam
activation	type de fonction d'activation	"relu" ou "linear"
patience	patience pour l'early_stopping	10
epochs	nombre d'époques	100
kernel_size	taille du noyau de convolution	3 / (6,2)

Nous avons entraîné les architectures en 1D et 2D en ajustant convenablement les dimensions des couches d'entrée et de sortie. Observons à présent les résultats obtenus sur le jeu de données test.

17. Les poids des neurones ayant été initialisés de façon aléatoire

18. Nombre d'instances d'entraînement (« samples ») sélectionnés aléatoirement

19. Lorsque le modèle s'ajuste trop précisément au jeu de données utilisé pour l'entraîner.

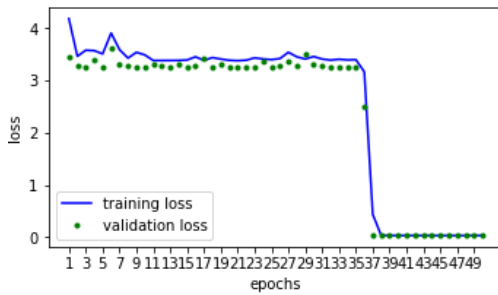
## 4.5.1 Régression

### 4.5.1.1 En 1D

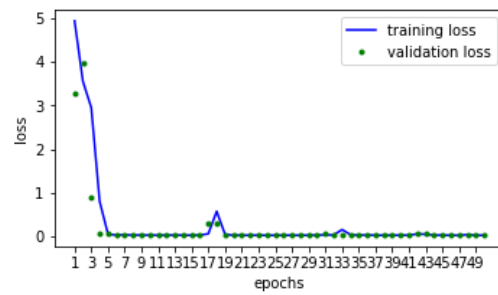
On obtient de très bonnes prédictions sur la hauteur de l'obstacle car celle-ci affecte directement l'amplitude des signaux sur le bord droit du domaine. Cela se traduit par un score  $R^2$  très élevé. Mais ce score n'est pas assez indicatif vu que les prédictions sur la position du créneau de densité ne sont pas bonnes<sup>20</sup>. Le score personnalisé permet de capturer ce défaut et donne environ 25 % (voir 4.2).

TABLE 4.2 – Résultats obtenus sur le jeu test en 1D. Le modèle avec Max-pooling correspond à la figure 4.6, et celui sans Max-pooling à la figure 4.7, tous deux implémentés pour des données 1D. Ces valeurs représentent une moyenne obtenue sur plusieurs cycles d'entraînement/prédiction.

Score	Avec Max-pooling	Sans Max-pooling
$R^2$	99.49 %	99.50 %
Personnalisé	26.50 %	28.21 %



(A) Avec Max-pooling



(B) Sans Max-pooling

FIGURE 4.12 – Comparaison de la vitesse de décroissance de la loss en 1D

Même si aucun n'est capable de correctement prédire la position du créneau, le modèle sans Max-pooling semble mieux se comporter sur notre jeu de données. Pour les illustrations qui vont suivre, nous utiliserons donc ce dernier modèle (sans Max-pooling). Nous commençons par une illustration de la corrélation entre les labels et les prédictions (figure 4.13).

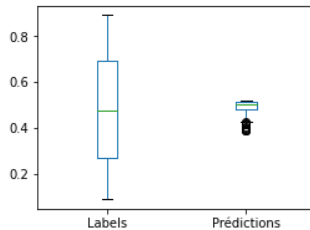
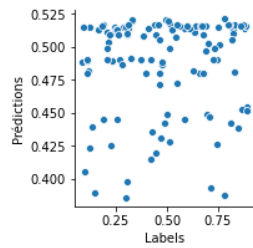
Pour observer les meilleures et les pires prédictions du modèle, il nous faut une mesure de la distance entre les prédictions et les labels. On définit donc la norme ci-bas (en prenant soins de normaliser la hauteur).

$$\text{Norme} = \sqrt{\text{Position}^2 + \left(\frac{\text{Hauteur}}{10}\right)^2}$$

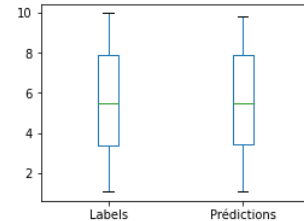
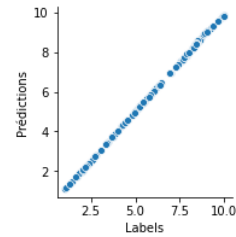
Observons donc les meilleures prédictions du modèle (sans Max-pooling) (figure 4.14).

Les pires prédictions du modèle permettent de mieux illustrer les problèmes rencontrés avec l'apprentissage 1D (figure 4.15).

20. Le score  $R^2$  élevé pourrait s'expliquer par le fait que le poids de la hauteur dans la corrélation est plus fort que celui de la position.

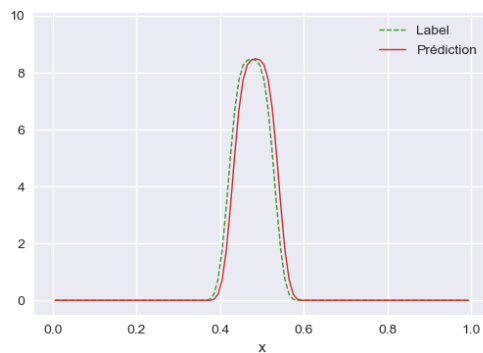


(A) Position

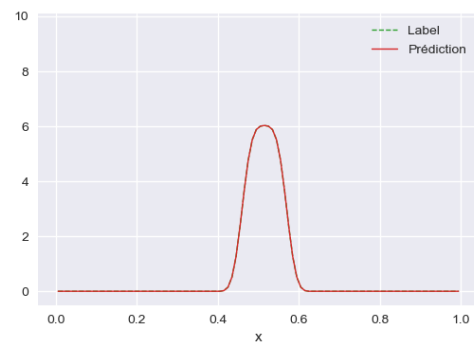


(B) Hauteur

FIGURE 4.13 – Illustration de la corrélation entre les labels et les prédictions obtenues par le modèle sans Max-pooling en 1D. On peut observer l'exactitude des prédictions pour la hauteur (B) mais un échec sur la position (A). En effet, les prédictions de la position du créneau sont concentrée autour de la moyenne 0.5.

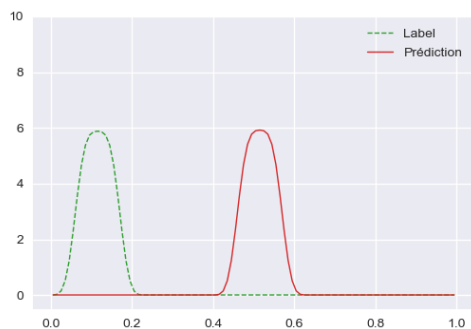


(A) Label = [0.488 8.474]  
Prédiction = [0.49 8.486]

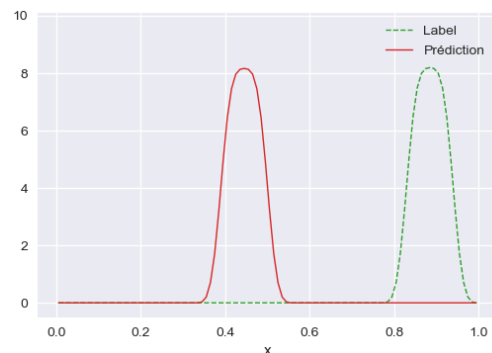


(B) Label = [0.522 6.032]  
Prédiction = [0.527 6.033]

FIGURE 4.14 – Les meilleures prédictions 1D. Il s'agit ici d'une reconstruction manuelle de la densité du milieu à partir des vecteurs en titre des figures (A) et (B). La première coordonnée indique la position du saut de densité, et la deuxième sa hauteur. On confirme que les bonnes prédictions des positions sont proches du milieu du domaine.



(A) Label = [0.125 5.886]  
Prédiction = [0.524 5.925]



(B) Label = [0.892 8.195]  
Prédiction = [0.454 8.167]

FIGURE 4.15 – Les pires prédictions du modèle 1D (sans Max-pooling). La différence se joue au niveau de la position du créneau comme indiquée à la figure 4.13. Les labels les plus éloignés du milieu conduisent aux prédictions les plus mauvaises.

La faible précision obtenue est due au fait que la simulation 1D est quasiment invariante par translation. En effet, une variation de la position du créneau de densité ne change les signaux sur la droite que de très peu. Ceci est probablement dû au fait que le problème inverse est mal posé, dans le sens où il n'y a pas unicité de la solution. Pour améliorer le score personnalisé, on pourrait passer en 2D.

#### 4.5.1.2 En 2D

En 1D, on ne mesurerait la sortie que sur 1 seul bord du domaine, ce qui limitait beaucoup l'apprentissage. En 2D, on récupère beaucoup plus d'informations (4 bords, 4 positions distinctes de la source) et cela renforce les chances d'unicité de la solution du problème inverse.

Comme attendu, le réseau est capable de détecter non seulement la hauteur de l'obstacle, mais aussi son abscisse et son ordonnée. Notre score personnalisé nous permet de confirmer cela dans le tableau 4.3.

TABLE 4.3 – Résultats obtenus sur le jeu test en 2D

Score	Avec Max-pooling	Sans Max-pooling
$R^2$	94.80 %	98.81 %
Personnalisé	55.75 %	93.50 %

On constate que le modèle sans l'opération de Max-pooling est globalement meilleur que son homologue avec Max-pooling, dû à l'application de l'Early-stopping. La figure 4.16 permet d'observer cela à travers la vitesse de convergence du modèle.

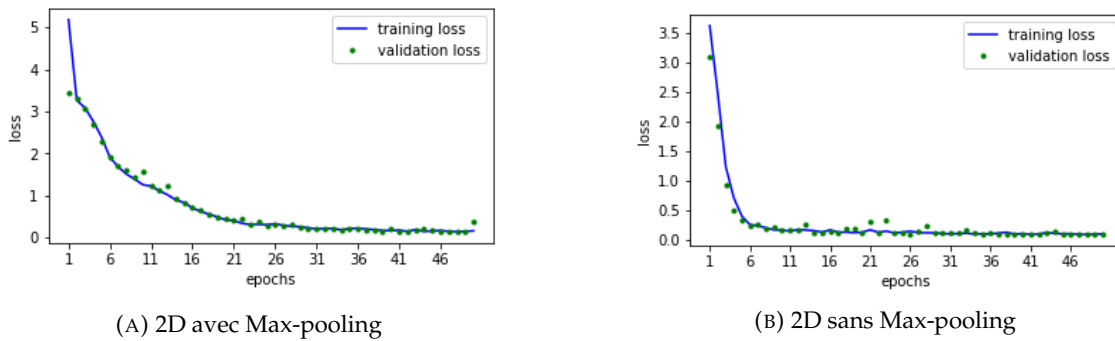


FIGURE 4.16 – Comparaison de la vitesse de décroissance de la loss en 2D

Comme nous l'avons fait en 1D, Le modèle sans Max-pooling sera utilisé par la suite pour illustrer la corrélation entre les prédictions (de l'abscisse  $x$ , de l'ordonnée  $y$ , et de la hauteur) et leurs labels respectifs (figure 4.17).

Pour observer les meilleures et les pires prédictions du modèle, on définit donc la norme ci-bas.

$$\text{Norme} = \sqrt{\text{Abcisse}^2 + \text{Ordonee}^2 + \left(\frac{\text{Hauteur}}{10}\right)^2}$$



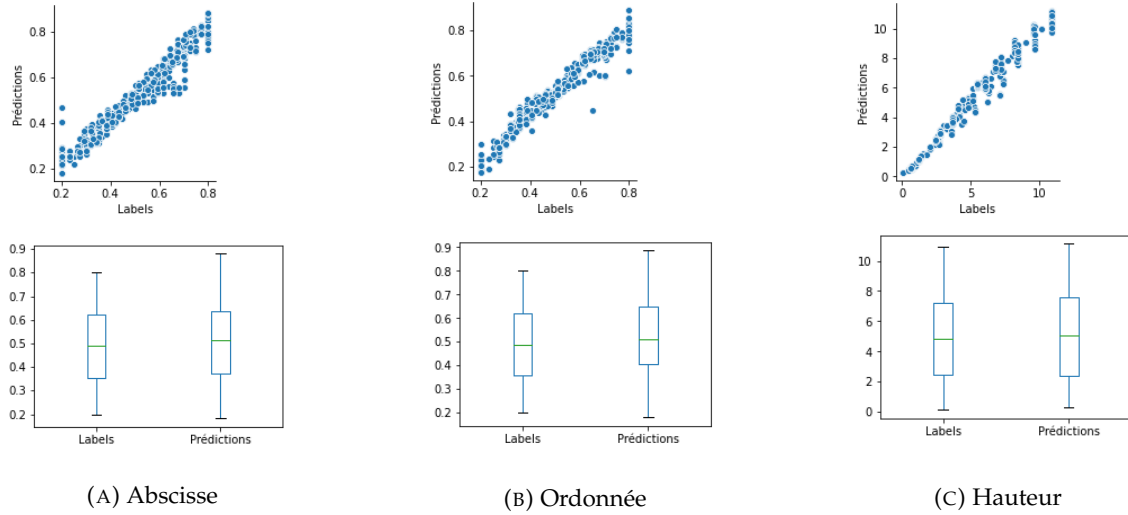


FIGURE 4.17 – Illustration de la corrélation entre les labels et les prédictions obtenues par le modèle sans Max-pooling en 2D. On peut observer que toutes les trois informations sont relativement bien corrélées, d'où le score  $R^2$  élevé (voir tableau 4.3).

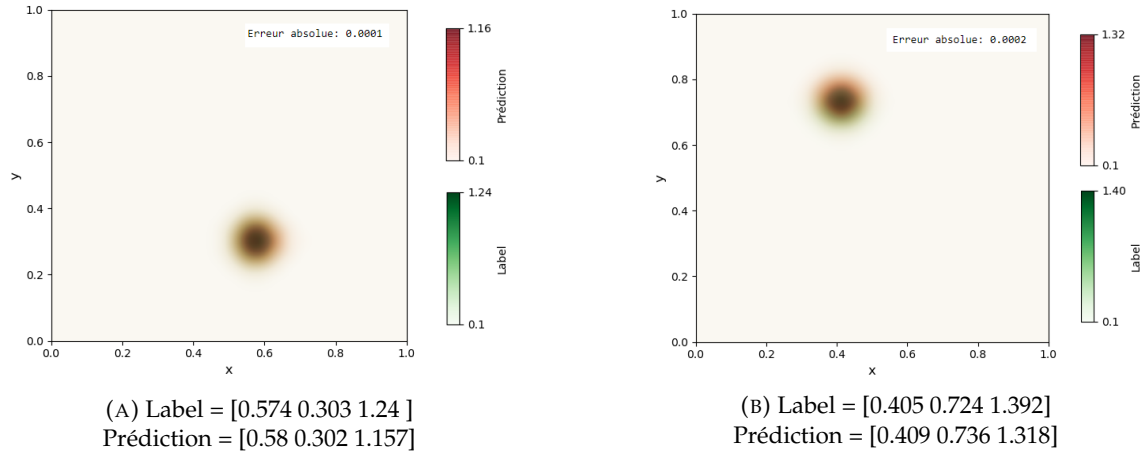


FIGURE 4.18 – Les meilleures prédictions 2D. Ces images ont été reconstruites à partir des vecteurs "Labels" et "Prédiction" issus de l'apprentissage. La première coordonnée indique l'abscisse  $x$  du saut de densité, la deuxième son ordonnée  $y$ , et la troisième sa hauteur. L'interpolation bicubique a été utilisée pour obtenir des images plus nettes.

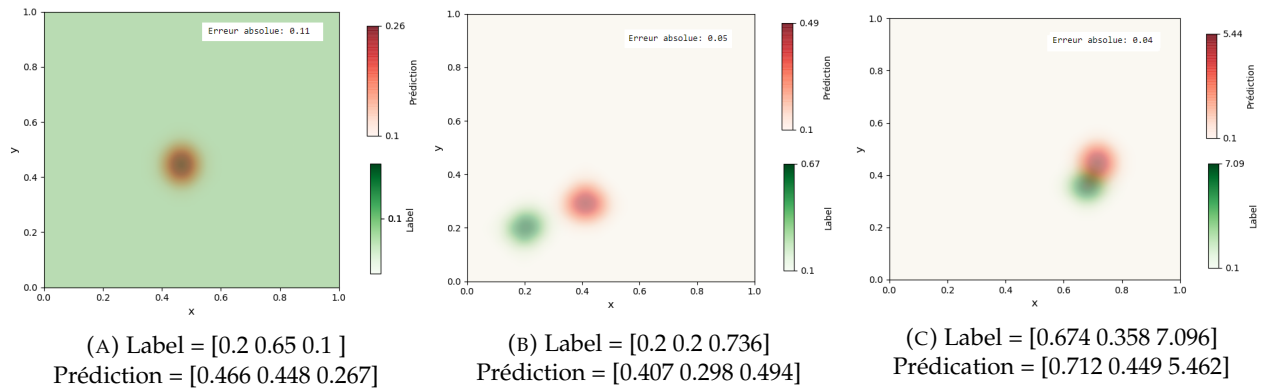


FIGURE 4.19 – Les pires prédictions du modèle 2D. Sans surprise la plus mauvaise des prédictions s'obtient lorsque le créneau est absent (A). En général, les pires prédictions sont faites lorsque le créneau se situe très proche de l'extrémité du domaine (B). La figure (C) représente un cas particulièrement mauvais de prédiction de la hauteur du créneau.

En ce qui concerne la généralisation du modèle à d'autres formes d'obstacles, je n'ai pas eu l'occasion de comparer les modèles avec et sans Max-pooling. Le modèle sans Max-pooling risque alors d'être moins performant conformément à la théorie (voir paragraphe 4.3.2). Sous Keras, le modèle à prouver être capable d'apprendre en continu, du moment que les entrées sont toutes normalisées et ont la même forme.

### 4.5.2 Classification

Une deuxième piste étudiée durant le stage a été la classification multilabel sur les données 2D. L'idée est de déterminer l'ordonnée de l'obstacle en trouvant en face de quelle(s) source(s) il est placé. La structure des entrées est quasiment la même que pour la régression en 2D (voir figure 4.3), sauf qu'il manque les signaux sur la gauche (coté où se trouve la source). En ce qui concerne les sorties, l'image ci-dessous décrit mieux leur structure :

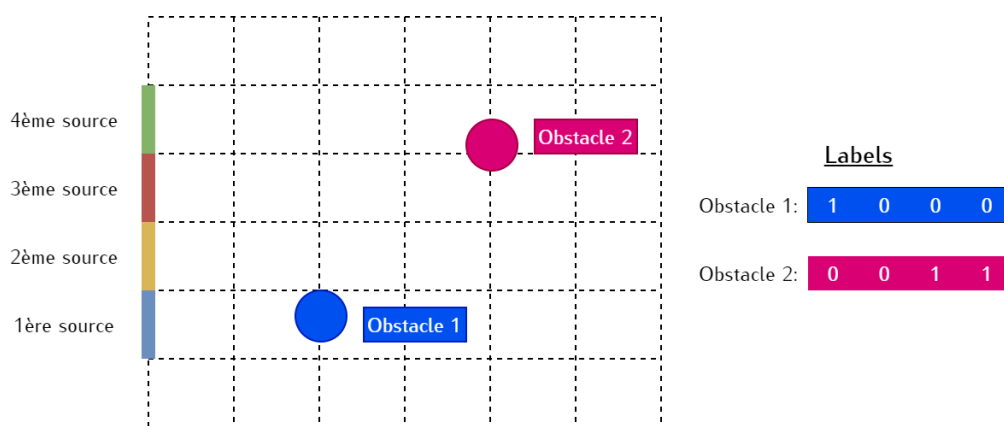


FIGURE 4.20 – Description des labels pour la classification multilabel en 2D. Un label est marqué 1 si l'obstacle se trouve dans le champ de la source correspondante, et 0 sinon.

Bien qu'ils aient une structure similaire, la taille des données utilisées pour la classification est différente de celle des données utilisées pour la régression (voir figure 4.4). On a moins d'itérations en temps (40 au lieu de 168) mais un maillage beaucoup plus fin (90x90 au lieu de 28x28). Le modèle de CNN utilisé ressemble beaucoup à celui utilisé pour la régression 2D. Une importante différence est qu'on utilise une activation "sigmoid" à la place de l'activation "linear" au niveau de la couche de sortie. On obtient donc en sortie des probabilités qu'il faut séparer en 2 classes suivant un seuil. Le modèle est entraîné avec des hyper-paramètres identiques à ceux utilisés durant la régression 2D. Les résultats sont présentés dans la table 4.4.

TABLE 4.4 – Résultats obtenus pour la classification en 2D. Le score sévère favorise les prédictions qui sont égales au véritable label sur toutes les 4 colonnes. Un seuillage convenablement effectué permet d'augmenter la précision des résultats. Le score de « Binary accuracy » procuré par Keras est calculé avant le seuillage.

Score	Avec Max-pooling	Sans Max-pooling
Binary accuracy	75.00 %	98.86 %
Score sévère après seuillage	27.27 %	95.45 %

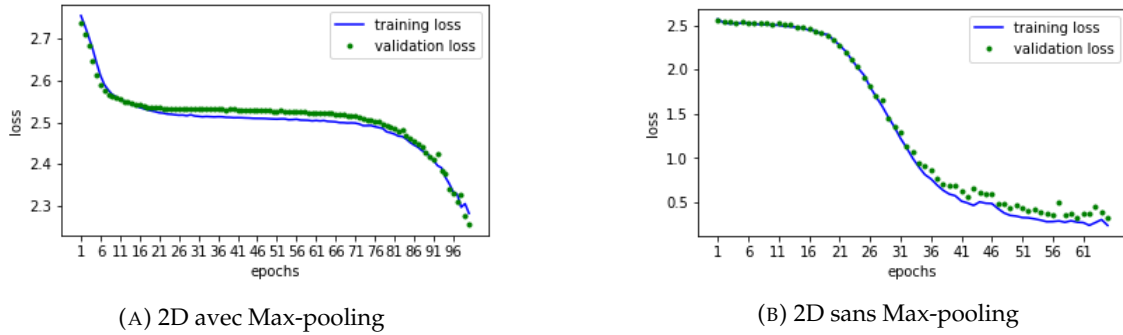


FIGURE 4.21 – Comparaison de la vitesse de décroissance de la loss (Multilabel Cross-entropy) pour la classification en 2D

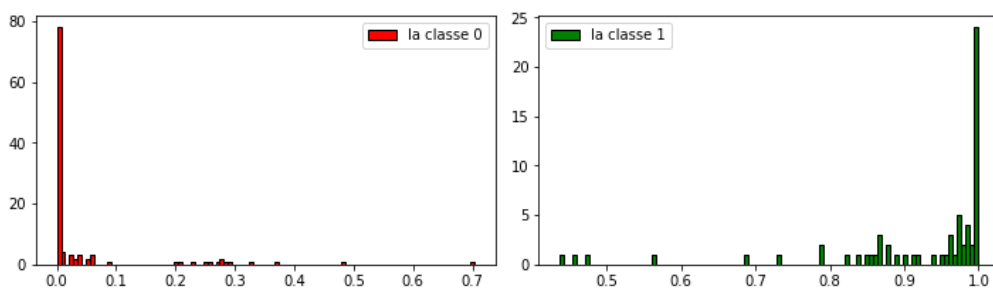


FIGURE 4.22 – Confiance du modèle sans Max-pooling en ses prédiction. En rouge la classe 0 qui indique l'absence du créneau devant la source, et en vert la classe 1. Cette figure indique que le modèle se trompe très rarement, ce qui confirme le résultat obtenu au tableau 4.4.

Dans ce cas aussi, le modèle sans Max-pooling semble meilleur (figures 4.21 et 4.22). Cependant nous ne disposons pas d'assez de données pour vérifier celui qui se généralise le mieux. Ceci étant une classification, les meilleures prédictions du modèle sont exactement les labels attendus et les pires prédictions ne diffèrent que de peu (de 1) de leurs cibles. Il est intéressant de constater qu'il n'y a aucune prédiction aberrante, par exemple : un obstacle se trouve en face des sources 1 et 3 sans reconstruire la source 2.

La classification est très prometteuse. Cependant elle n'offre pas beaucoup de précision car nous n'avons utilisé que quatre sources. Ceci constitue une grande limitation pour cette méthode, surtout dans des cas pratiques. En milieu médical par exemple, la localisation d'une tumeur se veut très précise. Pour remédier à ce problème, on pourrait augmenter le nombre de sources sur la gauche et en placer d'autres suivant la verticale. Il faudra penser à sous-échantillonner davantage les entrées car l'augmentation des sources est très coûteuse en espace mémoire (pour les données et pour les poids du modèle de CNN).

## 4.6 Conclusion sur l'apprentissage

Nous avons essentiellement effectué trois apprentissages durant le stage. Par ordre croissant de réussite, nous avons :

1. **Régression 1D** : Elle a permis de détecter la hauteur du créneau sur la densité d'un domaine 1D (avec la meilleure corrélation de tous les apprentissages). Elle n'a cependant pas

été capable de détecter la position du créneau, probablement dû au caractère mal posé du problème inverse.

2. **Classification 2D** : Elle a permis de localiser l'ordonnée du créneau en le situant par rapport aux sources sur la gauche d'un domaine 2D. En augmentant leur nombre et en plaçant certaines sources en haut (ou en bas) du domaine, on pourrait localiser avec plus de finesse l'abscisse et l'ordonnée du créneau.
3. **Régression 2D** : Elle a permis de prédire tous les attributs essentiels du créneau (abscisse, ordonnée, et hauteur), tout ceci avec une très forte précision (score personnalisé s'élevant à 93 %).

Durant chacun de ces apprentissages, nous avons comparé des modèles avec et sans Max-pooling en vue d'étudier la généralisation à d'autres formes d'obstacles et d'opacités. Le modèle sans Max-pooling a prouvé être supérieur dans le cas que nous avons à notre disposition.

## Chapitre 5

# Bilan du stage

### 5.1 Ressources utilisées

Les ressources utilisées durant le stage varient en nature et en fonction.

#### 5.1.1 Édition de code

- **Visual Studio Code (VSCode)** : Pour l'édition du code en langages C++. Ses nombreuses fonctionnalités, parmi lesquelles les extensions C/C++, CMake et CMake Tools ont considérablement facilité les travaux.
- **Google Colab** : Pour l'édition de code Python durant l'apprentissage. Les librairies les plus utilisées ici sont Numpy, Pandas, et Keras.
- **Jupyter** : Pour l'édition de code Python ne nécessitant pas trop de ressource (visualisation, sauvegarde en format PARQUET). Les librairies majeures utilisées ici sont Numpy, Pandas et Matplotlib.
- **Kile, VSCode, et Overleaf** : Pour l'écriture du rapport de stage en Latex. Kile a offert une très bonne rapidité de compilation. VSCode a permis d'accélérer la rédaction et surtout la. Overleaf a été efficace pour compiler en cas d'impossibilité de le faire en local.

#### 5.1.2 Simulations

Pour effectuer les simulations en grande quantité, nous nous sommes servi du serveur Atlas de l'Unistra. Dû à l'implémentation non parallèle du code C++, la vitesse des simulations était limitée. Mon PC a donc permis d'ajouter à cette puissance de calcul afin d'obtenir des jeux de données le plus rapidement possible.

#### 5.1.3 Apprentissage

Pour l'apprentissage, la puissance de calcul est venu uniquement de Google Colab. Ses GPU ont accéléré l'apprentissage de manière colossale, même si sa RAM de 12 Go a parfois été restreignante.

#### 5.1.4 Communication

Les communications se sont principalement effectuées par voie électronique (emails, Google Meet). J'ai aussi eu l'occasion de communiquer avec les professeurs en présentiel à trois reprises. Afin d'illustrer les informations communiquées (surtout dans le rapport), j'ai utilisé le logiciel **Draw.io**.

## 5.2 Journal de bord

### 5.2.1 Semaines 1 et 2

- 15 juin : Réunion de début de Stage par Google Meet.
- 16 juin : Demande aux professeurs de vérifier un exemple de simulation 1D, avant de me lancer la génération des données.
- 17 juin : Remarque du problème de formation et de propagation d'un créneau indésirable sur l'énergie, sur le flux et sur la température.
- 18 juin : Rédaction d'un nouveau schéma par M. Franck (pour l'étape 1) qui devrait conserver l'équilibre.
- 22 juin : Détection de la source du problème d'apparition du créneau indésirable, et définition d'un nouveau terme  $S'$ .
- 23 juin : Confirmation de l'exactitude des simulations 1D et début de la génération des données avec 500 mailles.
- 25 juin : Demande d'aide à M. Vigon pour la configuration de la fonction d'activation de la couche de sortie sous Keras.

### 5.2.2 Semaines 3 et 4

- 3 juillet : Rencontre avec M. Navoret pour discuter des avancements. Prise de connaissance d'une des raisons potentielles du problème de mauvaise prédiction de la position du créneau sur la densité en 1D. Proposition de plusieurs solutions par M. Navoret, entre autre de partir d'un signal stationnaire sinusoïdal et d'introduire l'onde à un temps  $t^* > 0$ .
- 6 juillet : Nouvelles simulations effectuées en vue d'observer la différence entre les effets de deux densités différentes. Continuation vers des nouvelles simulations avec 300 mailles.
- 8 juillet : Décroissance du taux d'apprentissage à la suggestion de M. Franck mais non-amélioration des résultats d'apprentissage.
- 9 juillet : Passage aux réseaux convolutif grâce à M. Vigon.
- 11 juillet : Plot du début des oscillations, des maximum, des minimum à la demande de M. Vigon, afin de mieux observer les effets de deux créneaux différents.

### 5.2.3 Semaines 5 et 6

- 13 juillet : Rencontre avec M. Navoret et M. Franck à la fac. Devant la persistance du problème de non-détection de la position du créneau, l'implémentation du problème en 2D semble être la solution appropriée.
- 14 juillet Reformulation 2D du schéma de « splitting » et adaptation du code 1D en 2D.
- 19 juillet : Fin du codage 2D et présentation des résultats.
- 25 juillet : Ajustement de la gamme de couleurs pour les visualisations et passage à la génération des données sur 90x90 mailles.

### 5.2.4 Semaines 7, 8, et 9

- 5 août : Rencontre avec M. Franck à la fac. Proposition de solutions pour la non-détection de la position du créneau en 2D par résolution d'un système proche de l'équation de la chaleur, après affichage par lignes de niveaux. La possibilité d'adopter un obstacle s'étendant sur toute la verticale est envisagée. Prise de connaissance des délais pour la rédaction du rapport de stage.
- 6 août : Rédaction et envoi du plan du rapport de stage.
- 7 août : Proposition de réduction drastique de la résolution spatiale, et proposition de nouvelles idées par M. Vigon, entre autre la considération d'un obstacle considérable plus opaque.

- 8 août : Nouvel apprentissage avec des simplifications majeures (entre autre un maillage 28x28) qui fonctionne. Amélioration des résultats et continuation du rapport.
- 15 août : Soumission du premier brouillon du rapport.

### 5.3 Difficultés rencontrées et solutions apportées

Un résumé du déroulement du stage est présenté à la figure 5.1. Les points les plus marquants du stage et les difficultés majeures rencontrées y sont représentés.

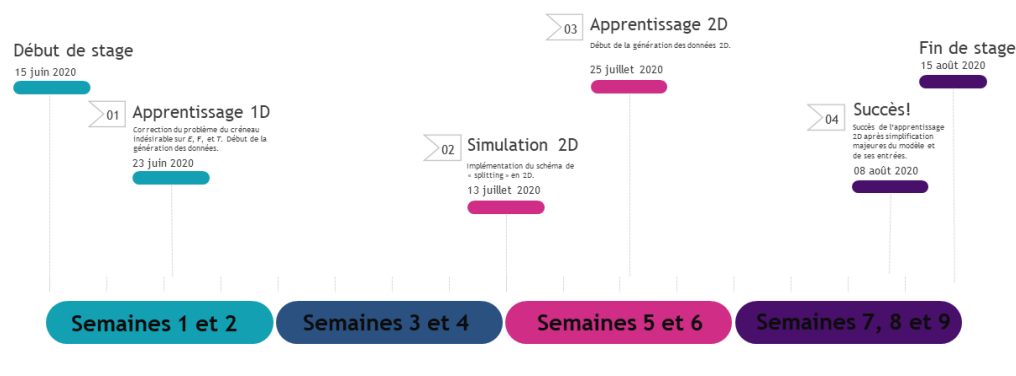


FIGURE 5.1 – Résumés des tournants du stage. Les détails du déroulement peuvent être obtenues dans la section 5.2.

#### 5.3.1 Apparition d'un créneau indésirable

Au commencement du stage, le code de calcul 1D n'était pas au point. En effet, dès qu'on plaçait un créneau sur la densité, un créneau correspondant se formait puis se propageait sur les signaux  $E$ ,  $F$ , et  $T$ . Le problème a été résolu par rajout d'un terme  $S'$  au niveau de la deuxième équation du schéma de « splitting ».

#### 5.3.2 Détection de la position du créneau

La détection de la position du saut de densité a été un problème récurrent durant le stage. À la fin du stage, aucune solution (si elle existe) n'a été trouvée pour le problème d'apprentissage en 1D. Cependant en 2D le problème a été résolu par diminution du taux d'apprentissage à  $1e-5$  et augmentation du nombre d'époques. Il est bien connu que l'entraînement des réseaux de neurones peu diverger si le taux d'apprentissage est trop élevé. Quant aux époques, je n'en faisais pas suffisamment pour voir le modèle converger. Une solution bien plus rapide aurait été d'automatiser la recherche des hyper-paramètres, chose que je n'ai apprise qu'à la fin du stage.

La solution trouvée en 2D, bien que très précise est loin d'être optimale pour une utilisation pratique (par exemple dans l'élaboration d'un tomographe). Elle souffre de deux problèmes majeurs :

- **Pas de généralisation** : le modèle finalement retenu ne comporte pas de couche de Max-pooling, une opération qui aurait probablement aidé à sa généralisation. Aucun test n'a été fait pour vérifier le niveau de généralisation de ce modèle.
- **Modèle trop lourd** : le nombre de paramètres est très élevé dû à la taille énorme des entrées, ce qui nécessite un espace mémoire très important au moment de la sauvegarde du modèle et de ses poids (2 Go).

Ces pistes d'amélioration n'ont pas pleinement été explorées pour cause de temps. En effet, la gestion du temps a été un vrai problème tout au long du stage.

### 5.3.3 Gestion du temps

La gestion du temps durant le stage n'a pas été facile. Au moment d'implémenter le schéma en 2D (ce qui n'était pas initialement prévu), j'ai longuement hésité sur l'option la plus rapide. J'ai pu compter sur les conseils de M. Navoret pour surmonter cet obstacle.

Une autre raison de mon retard est que je me suis rendu compte des délais bien en retard, j'ai dû me débrouiller pour terminer l'apprentissage et améliorer les résultats tout en rédigeant le rapport. Cela dit, je n'ai pas réussi à faire une partie essentielle qui consiste à étudier l'apport du Max-pooling dans la généralisation du modèle de CNN.

## 5.4 Les apports du stage

Ce stage a été enrichissant pour moi sur plusieurs fronts :

### 5.4.1 Expérience en développement

J'ai gagné de l'expérience en développement C++ et Python, tout en me développant un portfolio. J'ai beaucoup appris sur l'interface de programmation (API) de Pandas, de Matplotlib, et de Keras. J'ai à présent une large base de données de code réutilisable pour d'autres tâches.

### 5.4.2 Équations aux dérivées partielles

J'ai pu observer directement quelques astuces utilisées par MM. Navoret et Franck pour vérifier la validité d'une simulation. Sur l'équation du transfert radiatif, j'ai compris la nécessité de partir d'un état d'équilibre radiatif.

### 5.4.3 Réseaux de neurones

Ce stage m'a permis de percevoir la puissance des réseaux de neurones. J'ai appris à quel point le taux d'apprentissage est important, comme le rappelle cette citation tirée du livre de référence en apprentissage automatique *Deep Learning* : « *The learning rate is perhaps the most important hyperparameter. If you have time to tune only one hyperparameter, tune the learning rate.* » (GOODFELLOW et al., 2016, p. 417).

Cet enseignement, aussi clair qu'il est, me pousse néanmoins à me poser quelques questions concernant la place du « batch size ». Lors de l'apprentissage (régression 2D), il a fallu entraîner le modèle en utilisant la méthode d'**augmentation du « batch size »** pour obtenir les tous premiers résultats suffisamment bons pour être acceptables. Cette méthode, référencée [ici](#), montre que beaucoup de questions restent à résoudre dans le domaine de l'apprentissage profond.

### 5.4.4 Expérience de recherche

En tant que première expérience dans un environnement de recherche tel que l'IRMA, j'ai pu me familiariser avec le milieu. J'ai notamment appris que les résultats ne doivent pas toujours être ceux auxquels on s'attend, du moment que l'on a une explication pour l'échec.



## Chapitre 6

# Conclusion

Pour conclure, j'ai effectué mon stage de master 1 en tant que stagiaire en calcul scientifique et traitement de données à l'UFR de mathématiques et d'informatique de l'Université de Strasbourg. Lors de ce stage de deux mois, je me suis confronté au problème inverse de reconstruction de la densité d'un domaine par un CNN après avoir simulé la propagation d'un signal dans ce dernier en 2D. J'ai ainsi pu mettre en pratique mes connaissances (en calcul scientifique, en méthodes numériques, et en développement C++ et Python) acquises durant ma formation de CSMI.

En s'intéressant à un problème d'interaction entre l'intelligence artificielle (IA) et les EDP, ce stage m'a permis d'apprendre sur plusieurs plans. J'ai pu approfondir mon savoir sur les notions clés (« backpropagation », descente de gradient, taux d'apprentissage, etc.) utilisées dans les réseaux de neurones profonds. J'ai gagné beaucoup d'expérience en développement logiciel et je me suis familiarisé avec l'environnement de Keras. Ce stage m'a aussi permis de comprendre le déroulement d'une activité de recherche, et à quel point une bonne organisation et un certain degré d'autonomie sont importants. Un point sur lequel j'aurais voulu m'améliorer davantage est celui de la théorie des EDP, en particulier l'intuition derrière la définition des flux numériques.

Cette expérience de stage fut centrée autour de la problématique de l'apport des réseaux de neurones dans la résolution des problèmes inverses, applicable à la détection des tumeurs<sup>1</sup>. Le modèle de CNN construit est la preuve qu'un réseau de neurones est capable de reconstruire fidèlement des sauts de densité de nature cylindrique (ou circulaire vu du haut). L'utilisation du Max-pooling est sans doute un atout important pour la généralisation du modèle (détection de formes d'obstacles diverses, différentes expressions des opacités, etc.).

Fort de cette expérience et de ses nombreux enjeux, j'aimerais beaucoup par la suite, via un prochain stage, affiner les résultats à l'aide d'un apprentissage en continu en passant à la détection de plusieurs sauts de densités par exemple. On pourrait aussi envisager la reconstruction des opacités d'absorption et de dispersion. Tout ceci pourrait conduire ultimement à la création d'un tomographe et un déploiement en milieu médical.

---

1. Les tumeurs sont assimilables à des crâneaux, des sauts, ou des obstacles sur la densité du milieu.

## Annexe A

# Comment reproduire les resultats ?

### A.1 Exécution du code 1D/2D

Les codes de calculs 1D (développé durant le projet et utilise pour l'apprentissage durant ce stage) et 2D se trouvent sur deux dépôts Github différents :

- 1D : <https://github.com/desmond-rn/projet-inverse>
- 2D : <https://github.com/desmond-rn/projet-inverse-2d>

Pour compiler le code dans les deux cas, il faut idéalement passer par un conteneur Docker. Cependant la compilation peut aussi marcher si on passe par CMake directement. Ensuite il faut lancer l'exécutable "transfer" avec un fichier de configuration. Les détails ainsi que la définition des paramètres 1D/2D du fichier de configuration sont définis dans les fichiers README.md des dépôts 1D et 2D respectifs.

### A.2 Lecture du format binaire

Pour passer d'une simulation (C++) à sa visualisation ou à l'apprentissage (Python), les données peuvent être sauvegardées au format binaire à l'aide du paramètre "export\_type". Lorsque c'est le cas, il faut se servir de la fonction `read_sds_version01` (disponible dans le notebook principal de tout l'apprentissage [Regression.ipynb](#)) pour les lire.

Les données **train**, **val** et **test** que nous avons utilisé ont été sauvegardées sous le format binaire de Numpy. Ceci permet de faciliter la reproduction de l'apprentissage.

### A.3 Exécution des notebooks

Deux catégories de notebooks ont été créés dans les dépôts github. Les deux premiers sont exécutables directement après clonage du dépôts. Ils sont :

- Visualisation.ipynb : Il permet de visualiser les résultats d'une simulation exportée en CSV dans le fichier `data/df_simu.csv`.
- Sauvegarde.ipynb : Il permet de transformer des données du format CSV au format binaire PARQUET rapidement lisible par Pandas.

Les autres notebooks sont exécutables sur Google Colab. Ils ne sont exécutables que si on dispose des données train, val et test. On peut alternativement télécharger le modèle déjà entraîné depuis les dépôts Github.

## Annexe B

# Comment faire des prédictions avec ce modèle ?

Pour faire des prédictions, il suffit de disposer d'un jeu de données ayant la forme bien particulière décrite à l'une des figures 4.2 ou 4.4. Il faut ensuite charger le modèle à l'aide de Keras et ensuite le compiler.

### B.1 Normalisation des données

En plus d'avoir la forme appropriée, les données doivent être normalisées i.e toutes les énergies doivent être divisées par leur maximum (en valeur absolue). Il en est de même pour le flux et la température.

### B.2 Chargement du modèle

Le modèle a été sauvegardé sous la convention HDF5 de Keras. Après l'avoir chargé, il faut le compiler avec l'optimiseur Adam (et un taux d'apprentissage de  $1e-4$  ou  $1e-5$  suivant qu'on soit en 1D ou 2D). Pour que la compilation fonctionne, il faut impérativement inclure la fonction de calcul du score  $R^2$  indiquée ci-dessous.

```
from keras import backend as K

def r2_score(y_true, y_pred):
    SS_res = K.sum(K.square(y_true - y_pred), axis=-1)
    SS_tot = K.sum(K.square(y_true - K.mean(y_true)), axis=-1)
    return 1.0 - SS_res/(SS_tot + K.epsilon())
```

### B.3 Entraîner le modèle en continu

Le modèle peut être entraîné en continu. Après l'avoir chargé, on peut l'apprendre à détecter d'autres formes d'obstacles sous différentes conditions. Il suffit déjà de disposer de telles données. Pour l'apprentissage en continu, on pourra utiliser la fonction `train_on_batch` de Keras.

# Bibliographie

- FAUSKE, Kjell Magne (déc. 2006). « Example : Neural network ». In : URL : <https://texample.net/tikz/examples/neural-network/>.
- FRANCK, E. (oct. 2012). « Construction et analyse numérique de schéma asymptotic preserving sur maillages non structurés. Application au transport linéaire et aux systèmes de Friedrichs ». In : *Archive ouverte HAL*, p. 12, 160-161. URL : <https://www.nytimes.com/2015/12/11/science/an-advance-in-artificial-intelligence-rivals-human-vision-abilities.html>.
- GANESH, Prakhar (oct. 2019). « Types of Convolution Kernels : Simplified ». In : URL : <https://towardsdatascience.com/types-of-convolution-kernels-simplified-f040cb307c37>.
- GERON, Aurelien (2017). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow : Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc, p. 227-325. URL : [https://books.google.fr/books/about/Hands\\_On\\_Machine\\_Learning\\_with\\_Scikit\\_Le.html?id=HnetDwAAQBAJ&printsec=frontcover&source=kp\\_read\\_button&redir\\_esc=y](https://books.google.fr/books/about/Hands_On_Machine_Learning_with_Scikit_Le.html?id=HnetDwAAQBAJ&printsec=frontcover&source=kp_read_button&redir_esc=y).
- GOODFELLOW, Ian et al. (2016). *Deep Learning*. MIT Press, p. 5-6, 321-361, 417. URL : <http://www.deeplearningbook.org>.
- IRMA (mar. 2020). « Organigramme ». In : URL : <https://irma.math.unistra.fr/article816.html>.
- KERAS-TEAM (juin 2017). « Converting convolution kernels from Theano to TensorFlow and vice versa ». In : URL : <https://github.com/keras-team/keras/wiki/Converting-convolution-kernels-from-Theano-to-TensorFlow-and-vice-versa>.
- MARKOFF, J. (oct. 2015). « A Learning Advance in Artificial Intelligence Rivals Human Abilities ». In : *The New York Times*. URL : <https://www.nytimes.com/2015/12/11/science/an-advance-in-artificial-intelligence-rivals-human-vision-abilities.html>.
- PACKT (s.d). « Convolutional neural networks ». In : URL : [https://subscription.packtpub.com/book/game\\_development/9781789138139/4/ch04lv11sec31/convolutional-neural-networks](https://subscription.packtpub.com/book/game_development/9781789138139/4/ch04lv11sec31/convolutional-neural-networks).
- TURPAULT, R. (fév. 2004). « Modelisation, approximation numerique et applicationsdu transfert radiatif en desequilibre spectral couple avec l'hydrodynamique ». In : *Archive ouverte HAL*, p. 13-22. URL : <http://docplayer.fr/177010094-Modelisation-approximation-numerique-et-applications-du-transfert-radiatif-en-desequilibre-spectral-couple-avec-l-hydrodynamique.html>.