Université de Lille

Travail Encadré de Recherche

Interpolation par des fonctions radiales

Ayman Makki & Tarek Challioui

Dirigé par
Thomas Henneron

May 11, 2019

Contents

1 Introduction **4**

2 Construction de l'espace d'interpolation **6**
2.1 Fonctions radiales 6
2.2 Espace d'interpolation à base de fonction radiales 7
2.3 Théorème de Haar-Mairhuber-Curtis 8
2.4 Fonctions radiales définies positives 9
2.4.1 Théorème de Bochner 10
2.4.2 Fonctions complétement monotones 11
2.4.3 Théorème de Hausdorff-Bernstein-Widder 12
2.4.4 Fonctions à monotonicité mutliple 13

3 Interpolation à base de fonctions radiales **15**
3.1 Interpolation d'une fonction f connue 15
3.1.1 Première approche 16
3.1.2 Le choix de la fonction φ 17
3.1.3 Le choix du paramètre de forme ϵ 20
3.2 Optimisation de l'interpolation 22
3.2.1 Sélection d'un sous ensemble de $\mathcal{X}$ 22
3.2.2 Ajout d'un terme polynomial 23
3.3 Interpolation à partir d'un échantillon 25
3.3.1 Reconstruction d'un visage 25
3.3.2 Reconstruction d'un paysage volcanique 26

4 Application : résolution d'équations aux dérivées partielles **27**
4.1 Equation de poisson 27
4.2 Equation de la chaleur 30

5 Application : réseaux de neurones artificiels à base de fonctions radiales **32**
5.1 Notions fondamentales 32
5.2 Architecture d'un réseau de neurones à base de fonction radiales (NN-RBF) 33
5.3 Apprentissage par descente de gradient 34
5.4 Résultats 37

6 Conclusion **40**

List of Figures

1 Interpolation linéaire pour deux discrétisations différentes 4
2 Distinction entre approximation et interpolation 5
3 Fonctions radiales en une et deux dimensions ($r = \|x\|$) 6
4 Fonction ϕ_2 pour différentes normes sur $\mathbb{R}^d$ 7
5 Fonctions complètement monotones dans $\mathbb{R}$ 12
6 Fonctions complètement monotones dans $\mathbb{R}^2$ 14
7 φ dans $\mathbb{R}$ et $\mathbb{R}^2$ pour différentes valeurs de α 14
8 Deux exemples de fonctions à interpoler 15
9 Discrétisations des fonctions dans $\mathbb{R}$ et $\mathbb{R}^2$ 16
10 Interpolation de f_1 et de f_2 à l'aide de fonctions radiales gaussiennes 17
11 Décomposition de Φ_{f_1} pour deux paramètres de forme ϵ différents 17
12 Comparaison entre RBF et d'autres méthodes d'interpolation 18
13 Courbes des fonctions radiales dans $\mathbb{R}$ avec $\epsilon = 2$ 18
14 Surfaces de fonctions radiales dans $\mathbb{R}^2$ avec $\epsilon = 1$ 19
15 Comparaison des interpolations en fonction de φ et de ϵ 19
16 Deux interpolations de f_2 20
17 Influence du paramètre de forme sur les fonctions radiales dans $\mathbb{R}$ 21
18 Influence du paramètre de forme sur la fonction MQI dans $\mathbb{R}^2$ 21
19 Ecarts quadratiques moyens pour f_1 et f_2 en fonction de ϵ 21
20 Algorithme de sélection d'un sous ensemble de $\mathcal{X}$ avec différents critères de précision exigés (en rouge les points sélectionnés par l'algorithme) 22
21 Algorithme de sélection d'un sous ensemble de $\mathcal{X}$ avec différents critères de précision exigés (en rouge les points sélectionnés par l'algorithme) 22
22 Algorithme de sélection d'un sous ensemble de $\mathcal{X}$ avec différents critères de précision exigés (en rouge les points sélectionnés par l'algorithme) 23
23 Comparaison entre interpolation RBF et ajout de terme polynomial de différents degrés sur différents intervalles 24
24 Echantillon de points décrivant un visage humain 25
25 Interpolation sur un échantillon de 2628 points décrivant un visage humain 25
26 Echantillon de points décrivant un paysage volcanique 26
27 Interpolation sur un échantillon de 3216 points décrivant un paysage volcanique . . 26
28 Solution exacte u de l'équation de Poisson 27
29 Discrétisation aléatoire de $\mathcal{X}_\Omega$ (en vert) et $\mathcal{X}_{\delta\Omega}$ (en rouge) 28
30 Interpolation de l'équation de poisson sur un domaine rectangulaire 29
31 Variations des paramètres de formes et de la taille de la discrétisation 29
32 Erreur en valeur absolue pour une variation des paramètres de l'interpolation . . . 29
33 Discrétisation de l'espace pour l'équation parabolique 31
34 Solution de l'équation parabolique 31
35 Exemples de chiffres manuscrits 32
36 Architecture d'un réseau de neurones à base de fonctions radiales 34
37 Mécanisme d'apprentissage d'un réseau de neurones 34
38 Influence du taux d'apprentissage dans la minimisation de la fonction de coût . . . 35
39 Performances du réseau de neurones à base de fonctions radiales 37

List of Tables

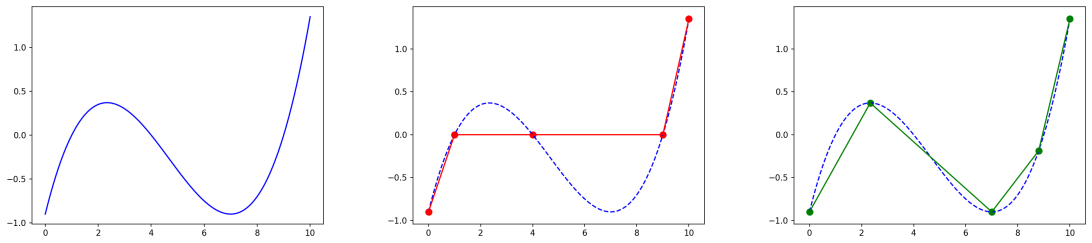
1 Solution du système linéaire pour f_1 16
2 Exemples de fonctions associées à des fonctions radiales définies positives 18
3 EQM(Φ_{f_1}) en fonction de φ et de ϵ (en gras : minimum par ligne) 19
4 EQM(Φ_{f_2}) en fonction de φ et de ϵ (en gras : minimum par ligne) 20
5 Comparaison des EQM et du conditionnement de la matrice 24
6 Exemples de fonctions de coût pour un problème de classification 33
7 Résultats du réseau de neurones avec 5000 itérations 37
8 Résultats du réseau de neurones RBF avec ACP sur 61 composantes et 10 000 itérations 38

1 Introduction

La continuité est une notion à la fois fortement intuitive et incroyablement complexe. D'un côté, l'ensemble de nos perceptions nous semble parfaitement continu, qu'il s'agisse de l'espace qui nous entoure, du temps qui s'écoule ou de nos capacités auditives et olfactives par exemple. Nous ne percevons pas l'instant en lui même, mais un continuum que l'on pourrait voir comme une infinité d'instants infiniment petits. C'est d'ailleurs sur cette conception de la continuité, constituée de variations infinitésimales, que sont définies les notions mathématiques de continuité de fonctions.

Malgré cette intuition universelle et communément acceptée de la continuité, il est incroyablement complexe - impossible pour l'instant - de l'étudier en tant que telle lorsque que l'on s'intéresse à des phénomènes dans des espaces continus. Premièrement, nous avons des contraintes liées à la mesure : qu'il s'agisse du temps ou de l'espace, nous ne savons effectuer des mesures en chaque instant, en chaque point de l'espace. Deuxièmement, dans le cas ou nous saurions effectuer des mesures continues, il ne serait pas possible de les enregistrer : les technologies actuelles de stockage de masse basées sur des matériaux paramagnétiques ou sur des transistors ne peuvent servir à encoder une information continue : elle doit être discrétisée, c'est à dire réduite à certains points de l'espace.

Il est assez évident de comprendre qu'une discrétisation induit une certaine perte d'information : nous ne disposons pas de toute l'information qui caractérise le phénomène, puisqu'il n'est mesuré qu'en certains points de l'espace (temporel ou spatial), et il peut apparaître compliqué de le reconstruire numériquement. C'est ce que nous montrons dans la Figure 1 : on interpole linéairement une fonction donnée pour deux discrétisations de 5 points dont 3 diffèrent et on observe que dans le cas (b) l'interpolation est très médiocre alors qu'elle est bien meilleure dans le cas (c). Nous aborderons par ailleurs quelques techniques de sélection des points optimaux dans un ensemble de mesures donné pour nos problèmes d'interpolation.



(a) Fonction originale (b) Mauvaise discrétisation (c) Bonne discrétisation

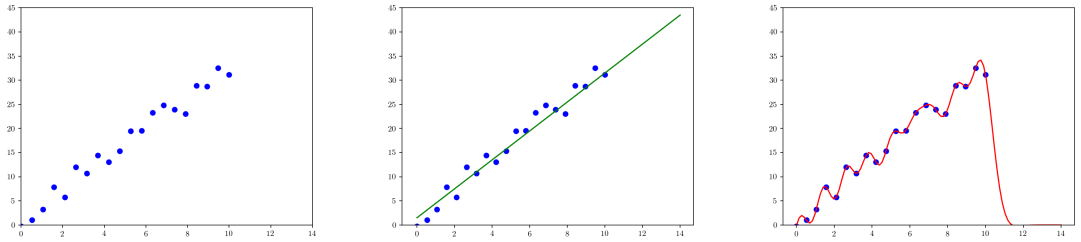
Figure 1: Interpolation linéaire pour deux discrétisations différentes

C'est dans ce cadre qu'interviennent les différentes théories mathématiques de l'interpolation et de l'approximation : il s'agit d'être capable, à partir d'un ensemble de points discrets de l'espace étudié et des valeurs mesurées en ces points, d'être capable de reconstruire une fonction avec certaines propriétés de continuité qui se rapproche de celle qui a permis de générer les mesures à notre disposition.

Par exemple, supposons que l'on veut étudier la température dans une région à un moment précis en ayant à notre disposition les relevés de température dans chacune des villes. L'interpolation nous permettra d'étudier les variations dans l'espace de la température entre deux villes, ou l'on ne dispose d'aucune information *à priori*. On peut aussi s'intéresser à la température dans une ville au cours du temps : ayant effectué des relevés toutes les heures pendant plusieurs années, on pourra essayer de prédire la température pour le lendemain à l'aide de techniques d'approximation.

On distingue alors deux approches, illustrées dans la Figure 2 : l'interpolation et l'approximation.

L'interpolation impose à la fonction de prendre les valeurs exactes mesurées, ce qui peut permettre de mieux observer ce qui se passe entre deux points (entre deux villes de la région) mais produit en général des résultats peu convaincants en dehors du domaine d'étude. L'approximation quant à elle recherche la minimisation d'un critère d'erreur, comme la distance entre la fonction reconstruite et les points mesurés, et permet en général une meilleur extrapolation à l'extérieur du domaine étudié (comme une journée future ou aucune mesure n'a été effectuée). Ci dessous, la figure (b) montre que l'approximation décrit mieux la tendance lorsqu'on étend le domaine alors que l'interpolation exacte aux points va juste lisser la fonction vers 0.



(a) Points mesurés (b) Approximation linéaire (c) Interpolation RBF

Figure 2: Distinction entre approximation et interpolation

Dans ce Travail Encadré de Recherche, nous nous intéresserons exclusivement à l'interpolation de fonctions. Il existe de nombreuses techniques d'interpolation à ce jour : interpolation polynomiale dont la plus classique est l'interpolation lagrangienne, mise au point par Edward Waring [War79], interpolation bilinéaire pour le redimensionnement d'images ou encore l'interpolation de Barnes pour [Bar64] pour la prévision climatique. Nous nous intéresserons plus particulièrement à l'interpolation à base de fonctions radiales (*Radial Basis Function, i.e RBF*) dans nos travaux et étudierons son utilité dans différents domaines d'application.

Dans un premier temps sera étudiée la construction théorique de l'interpolation par des fonctions radiales afin de pouvoir s'assurer de l'existence d'une solution au problème d'interpolation. Nous partirons de la définition des fonctions radiales et de leur caractéristiques et nous construirons l'espace d'approximation fonctionnelle dans lequel nous interpolerons la solution approchée.

Ensuite, nous étudierons l'interpolation à base de fonctions radiales à l'aide d'exemples simples basés sur des fonctions connues dans $\mathbb{R}$ et dans $\mathbb{R}^2$. Nous illustrerons chaque exemple et introduirons les différents paramètres de l'interpolation à base de fonctions radiales à l'aide de ces exemples. Nous aborderons entre autre le choix de la fonction radiale, du paramètre de forme, la sélection d'un sous-échantillon ou encore l'ajout d'un terme polynomial.

Finalement, nous proposerons deux applications des techniques d'interpolations à base de fonctions radiales : la résolution d'équations aux dérivées partielles et la reconnaissance des chiffres manuscrits à l'aide de réseaux de neurones.

2 Construction de l'espace d'interpolation

Dans l'ensemble des parties qui suivent, nous partons du principe que nous étudions un phénomène donné (variations de température, orientation politique d'une population) qui peut se modéliser comme une fonction f allant de l'espace vectoriel $\mathbb{R}^d$, muni de la norme euclidienne $\|.\|$, dans $\mathbb{R}$, avec $d \in \mathbb{N}^*$. On dispose alors d'un ensemble de points d'interpolation $\mathcal{X} = (x_1, \dots, x_n) \subset \mathbb{R}^d$ ou des mesures du phénomène ont effectuées et on note $\mathcal{Y} = (y_1, \dots, y_n) \subset \mathbb{R}$ l'ensemble des mesures connues *à priori*. On a alors $\forall i \in [\![1, n]\!],\ f(x_i) = y_i$.

L'objectif de l'interpolation est de pouvoir construire une fonction Φ approchant au mieux la fonction f, en lui imposant la contrainte de prendre la valeur mesurée en chacun des points d'interpolation, c'est à dire : $\forall i \in [\![1, n]\!],\ \Phi(x_i) = f(x_i) = y_i$. Dans le cadre théorique, f étant une fonction quelconque, elle appartient à un espace de dimension infinie. La première étape de notre recherche consiste à construire un espace d'approximation de dimension finie $\mathcal{V}$ dans lequel on va approcher f. Après avoir défini la notion de fonction radiale, nous nous pencherons sur la construction d'un tel espace et nous en déduirons les contraintes qui lui sont inhérentes afin que le problème d'interpolation soit bien posé, c'est à dire que l'on ait existence et unicité de la solution.

2.1 Fonctions radiales

Intuitivement parlant, une fonction radiale centrée en $x_0 \in \mathbb{R}^d$ est une fonction $\phi : \mathbb{R}^d \to \mathbb{R}$ qui prend sa valeur en fonction de la distance entre le point où on l'évalue et son centre x_0. Cela se traduit mathématiquement parlant par le fait qu'elle soit invariante pour toutes les rotations dans $\mathbb{R}^d$, ce qui induit la définition suivante, dans le cas ou $x_0 = 0_{\mathbb{R}^d}$:

Définition 2.1. Une fonction $\phi : \mathbb{R}^d \to \mathbb{R}$ est dite radiale si elle est invariante pour toutes les rotations conservant l'origine, c'est à dire :

$$\forall \rho \in SO(d),\ \phi = \phi \circ \rho$$

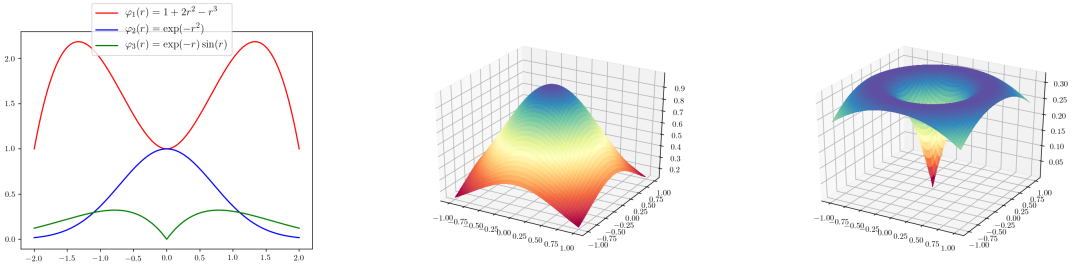
où $SO(d)$ représente le groupe orthogonal spécial de dimension d sur $\mathbb{R}$, c'est à dire l'ensemble des transformation linéaires orthogonales de déterminant égal à 1. Il s'agit de l'ensemble des rotations dans $\mathbb{R}^d$ qui préservent l'origine.

Remarque. La définition ci-dessus s'étend à toute fonction radiale centrée en un point x_0 différent de l'origine. Il suffit de considérer les rotations autour de ce point.

Puisqu'une fonction radiale ne dépend que de son centre et de la distance à ce dernier, on utilisera successivement les notations suivantes : soit ϕ une fonction radiale centrée en $x_0 \in \mathbb{R}^d$ et $x \in \mathbb{R}^d$, on note $\varphi : \mathbb{R} \to \mathbb{R}$ sa fonction réelle associée telle que :

$$\phi_{x_0}(x) = \varphi(\|x - x_0\|)$$

Voici quelques exemples de fonctions radiales centrées en 0 pour la norme euclidienne :



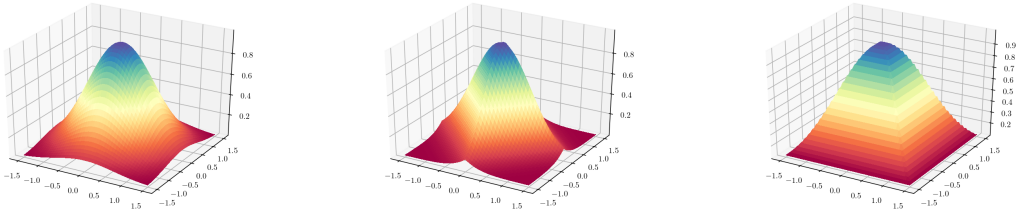
(a) Fonctions radiales dans $\mathbb{R}$ (b) φ_2 dans $\mathbb{R}^2$ (c) φ_3 dans $\mathbb{R}^2$

Figure 3: Fonctions radiales en une et deux dimensions ($r = \|x\|$)

Proposition 2.1. *Soit ϕ une fonction radiale centrée en x_0, alors les lignes de niveau de ϕ_{x_0} forment des cercles centrés en x_0. C'est à dire que, pour toute valeur k prise par la fonction ϕ_{x_0} en un point $x \in \mathbb{R}^d$, l'ensemble $\{y \in \mathbb{R}^d, \phi_{x_0}(y) = k\}$ peut s'écrire comme une union de cercles centrés en x_0.*

Preuve. Soit $x \in \mathbb{R}^d$, on note $k = \phi_{x_0}(x)$. On prend $\rho \in SO(d)$ une rotation de $\mathbb{R}^d$ préservant x_0. On a alors $\phi_{x_0}(\rho(x)) = \phi_{x_0}(x) = k$ et ce pour toutes les rotations préservant x_0. Ainsi, on note $K = \{\rho(x), \rho \in SO(d)$ telle que $\rho(x_0) = x_0\}$ l'ensemble de niveau rattaché à la valeur k qui contient x, et on remarque qu'il peut aussi s'écrire sous la forme $K = \{y \in \mathbb{R}^d, \|y - x_0\| = \|x - x_0\|\}$, puisque ρ est une isométrie et conserve le point x_0, ce qui décrit un cercle centré en x_0 et de rayon $\|x - x_0\|$. S'il existe un autre $x' \notin K$ tel que $\phi_{x_0}(x') = k$ alors cela décrit un autre cercle de rayon différent. Dans tous les cas, on pourra décrire l'ensemble des points ou ϕ_{x_0} prend la valeur k comme une union de cercles centrés en x_0.

D'après la proposition précédente, on déduit que la forme des fonctions radiales est dépendante de la norme utilisée. En effet, si on reprend la fonction $\phi_2(x) = e^{-\|x\|^2}$ de la figure précédente et qu'on modifie la norme, on obtient :



(a) Norme euclidienne $\|.\|_2$ (b) Norme 1 $\|.\|_1$ (c) Norme infinie $\|.\|_\infty$

Figure 4: Fonction ϕ_2 pour différentes normes sur $\mathbb{R}^d$

2.2 Espace d'interpolation à base de fonction radiales

On s'intéresse dans cette partie plus particulièrement à l'espace $\mathcal{V}$ de dimension finie dans lequel on va interpoler la fonction f. Puisqu'on interpole à l'aide de fonctions radiales, on peut noter $\mathcal{V} = \text{vect}(\phi_1, \ldots, \phi_N)$ l'espace fonctionnel engendré par N fonctions radiales. Ainsi, notre approximation dans cette base de fonctions radiales s'écrit sous la forme :

$$\Phi = \sum_{i=1}^{N} \alpha_i \phi_i$$

On conserve également les notations $(\mathcal{X}, \mathcal{Y})$ pour les points mesurés de notre fonction f. Dans un premier temps, on propose une première définition générale du problème d'interpolation dans une base de fonctions radiales :

Définition 2.2. On définit une solution du problème d'interpolation $(\mathcal{X}, \mathcal{Y}, \mathcal{V})$ de f dans $\mathcal{V}$ tout élément Φ de $\mathcal{V}$ tel que $\Phi_{|\mathcal{X}} = f_{|\mathcal{X}}$. Autrement dit, $\Phi \in \mathcal{V}$ est solution du problème d'interpolation de f dans $\mathcal{V}$ si

$$\forall x \in \mathcal{X},\ \Phi(x) = f(x)$$

Il nous intéresse plus particulièrement de construire $\mathcal{V}$ l'espace d'approximation afin de garantir l'existence d'un unique interpolant, quels que soient les points mesurés $(\mathcal{X}, \mathcal{Y})$. Nous montrerons d'abord qu'il n'est pas possible d'obtenir un tel espace pour avoir un problème d'interpolation bien posé quels que soient les points $(\mathcal{X}, \mathcal{Y})$, mais qu'il est possible, pour un ensemble de points donnés, de construire un espace $\mathcal{V}$ dépendant de ces points dans lequel notre solution existera et sera unique.

Définition 2.3 (Matrice d'interpolation)**.** On définit la matrice d'interpolation K de l'espace $\mathcal{V}$ telle que, pour un ensemble de points $\mathcal{X} \subset \mathbb{R}^d$, $K_{i,j} = \phi_j(x_i)$. $K_\mathcal{X}$ est donc une matrice de taille $(\text{card}(\mathcal{X}), \dim(\mathcal{V})) = (n, N)$ et elle s'écrit :

$$K = \begin{pmatrix} \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_N(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \phi_{N-1}(x_{n-1}) & \phi_N(x_{n-1}) \\ \phi_1(x_n) & \cdots & \phi_{N-1}(x_n) & \phi_N(x_n) \end{pmatrix}$$

Ainsi, nous pouvons exprimer notre problème d'interpolation comme la résolution d'un système linéaire dans $\mathbb{R}^N$:

Proposition 2.2. *Trouver une solution au problème d'interpolation de f dans $\mathcal{V}$ revient à résoudre le système linéaire suivant :*

$$K\alpha = Y \iff \begin{pmatrix} \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_N(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \cdots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & \cdots & \phi_{N-1}(x_{n-1}) & \phi_N(x_{n-1}) \\ \phi_1(x_n) & \cdots & \phi_{N-1}(x_n) & \phi_N(x_n) \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{N-1} \\ \alpha_N \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{pmatrix}$$

Il apparaît immédiatement qu'un vecteur $\alpha \in \mathbb{R}^N$ qui vérifierait le système ci-dessus garantirait à la fonction $\Phi = \sum_{i=1}^{N} \alpha_i \phi_i$ d'être solution du problème d'interpolation. Cependant, puisque nous voulons un problème bien posé, on souhaite s'assurer de **l'existence et de l'unicité** de cette solution en se servant des résultats de l'algèbre linéaire. Nous aimerions donc savoir si la matrice est inversible, ce qui nous impose une première contrainte sur notre espace d'approximation $\mathcal{V}$.

Théorème 2.3. *Soit $P = (\mathcal{X}, \mathcal{Y}, \mathcal{V})$ un problème d'interpolation. Si P admet une unique solution, alors card($\mathcal{X}$) = dim($\mathcal{V}$).*

Preuve. De manière assez triviale, on suppose $P = (\mathcal{X}, \mathcal{Y}, \mathcal{V})$ un problème d'interpolation admettant une unique solution et $K_\mathcal{X}$ la matrice d'interpolation du problème. Comme nous l'indique la Proposition 2.2, une solution au problème d'interpolation est une solution du système linéaire $K_\mathcal{X}\alpha = Y$.
Si P admet une unique solution, alors la matrice $K_\mathcal{X}$ est inversible, ce qui indique que c'est une matrice carrée, donc $n = N \iff \text{card}(\mathcal{X}) = \dim(\mathcal{V})$.

On obtient donc une première contrainte sur notre espace d'interpolation $\mathcal{V}$ qui nous garantit l'unicité de la solution mais pas son existence. Afin de prouver que l'existence de la solution ne peut se faire dans une base de fonctions radiales $\mathcal{V}$ quelconque indépendante des points $\mathcal{X}$, nous allons nous servir du théorème de Mairhuber-Curtis sur les espaces de Haar.

2.3 Théorème de Haar-Mairhuber-Curtis

Cette partie est inspirée de [BW03] et [Mar18] en grande partie. On commence par définir ce qui serait un espace d'interpolation *idéal*, dans le sens ou il permettrait d'interpoler sur tout ensemble de points $(\mathcal{X}, \mathcal{Y})$ dans un domaine donné. C'est ce qu'on appelle un espace de Haar.

Définition 2.4 (Espace de Haar)**.** Soit $\mathcal{V}$ un espace vectoriel fonctionnel de dimension n engendré par une base de fonctions radiales $(\phi_1, \ldots, \phi_n)$ définies sur $\mathbb{R}^d$. Soit K la matrice d'interpolation associée à $\mathcal{V}$. Alors $\mathcal{V}$ est un espace de Haar si

$$\det(K_\mathcal{X}) \neq 0, \ \forall \mathcal{X} = (x_1, \ldots, x_n) \subset \mathbb{R}^d$$

Comme nous avons vu précédemment, l'inversibilité de la matrice d'interpolation K garantit l'existence et l'unicité de l'interpolant dans la base $\mathcal{V}$. Ainsi, si nous pouvions construire un espace d'approximation $\mathcal{V}$ qui serait un espace de Haar, alors notre problème serait bien posé.

Exemple. Dans $\mathbb{R}$, l'espace des polynômes de degré $N-1$ forment un espace de Haar de dimension N.

Théorème 2.4 (Haar-Mairhuber-Curtis)**.** *Soit $d \geq 2$, et $\Omega \subset \mathbb{R}^d$ contenant un point intérieur, alors on ne peut construire un espace de Haar de fonctions continues sur Ω de dimension supérieure ou égale à 2.*

Preuve. Prenons $d, N \geq 2$ et supposons que $\mathcal{V} = \text{vect}(\phi_1, \ldots, \phi_n)$ est un espace de Haar sur $\Omega \subset \mathbb{R}^d$ où Ω contient un point intérieur.
Par l'existence du point intérieur, on peut trouver x_0 et r tels que la boule $B(x_0, r) \subset \Omega$ et on peut fixer $(x_3, \ldots, x_n) \in B(x_0, r)$ distincts deux à deux. Ensuite, on choisit deux courbes $x_1(t)$ et $x_2(t)$ pour $t \in [0, 1]$ telles que $x_1(0) = x_2(1)$ et $x_1(1) = x_2(0)$ tout en s'assurant qu'elles n'ont aucun autre point d'intersection entre elles ou avec $(x_3, \ldots, x_n)$, ce qui est possible car $d \geq 2$.
Puisque $\mathcal{V}$ est supposé être un espace de Haar sur Ω alors la fonction

$$D(t) = \det\left((\phi_j(x_i(t)))_{i,j \leq n}\right)$$

est continue et ne change pas de signe. Or, $D(1) = -D(0)$ car cela revient juste à échanger les deux premières lignes. Donc D est nulle pour un certain $t^* \in [0, 1]$ et donc $\mathcal{V}$ n'est pas un espace de Haar.

Ainsi, on ne peut construire $\mathcal{V}$ comme un espace de Haar dès que $\mathcal{X} \subset \mathbb{R}^2$ contient au moins deux points, car nous aurons besoin que la dimensions de $\mathcal{V}$ soit égale à deux, comme vu précédemment. Ne pouvant construire un espace d'approximation $\mathcal{V}$ fonctionnant pour tout ensemble de points $(\mathcal{X}, \mathcal{Y})$, on se fixe un ensemble de points et on cherche à définir $\mathcal{V}$ de telle sorte que la matrice $K_{\mathcal{X}}$ associée soit inversible.

2.4 Fonctions radiales définies positives

On cherche donc à définir $\mathcal{V}$ en fonction de $\mathcal{X}$ de telle sorte à ce que l'on puisse garantir que la matrice d'interpolation $K_{\mathcal{X}}$ soit inversible. On sait que notre espace d'interpolation $\mathcal{V}$ doit être de dimension égale au nombre de points contenus dans $\mathcal{X}$. Puisque nous avons vu que les fonctions radiales sont définies autour d'un centre x_0, nous souhaitons prendre les points de $\mathcal{X}$ comme centres des fonctions radiales qui génèrent $\mathcal{X}$. Ainsi, nous avons $\mathcal{V} = \text{vect}(\phi_{x_1}, \ldots, \phi_{x_n})$ et la matrice d'interpolation $K_{\mathcal{X}}$ s'écrit dorénavant :

$$K = \begin{pmatrix} \phi_{x_1}(x_1) & \phi_{x_2}(x_1) & \cdots & \phi_{x_n}(x_1) \\ \phi_{x_1}(x_2) & \phi_{x_2}(x_2) & \cdots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & \cdots & \phi_{x_{n-1}}(x_{n-1}) & \phi_{x_n}(x_{n-1}) \\ \phi_{x_1}(x_n) & \cdots & \phi_{x_{n-1}}(x_n) & \phi_{x_n}(x_n) \end{pmatrix}$$

ce qui permet d'obtenir une matrice symétrique grâce à la symétrie de la norme :

$$K = \begin{pmatrix} \phi(0) & \phi(\|x_1 - x_2\|) & \cdots & \phi(\|x_1 - x_n\|) \\ \phi(\|x_2 - x_1\|) & \phi(0) & \cdots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & \cdots & \phi(0) & \phi(\|x_{n-1} - x_n\|) \\ \phi(\|x_1 - x_n\|) & \cdots & \phi(\|x_n - x_{n-1}\|) & \phi(0) \end{pmatrix}$$

On peut donc se servir de tous les résultats sur les matrices symétriques afin d'étudier l'inversibilité de la matrice $K_{\mathcal{X}}$ et donc l'existence est l'unicité de la solution à notre problème d'interpolation.

Définition 2.5. Une matrice symétrique réelle $K \in \mathcal{M}_n(\mathbb{R})$ est dite semi-définie positive si, $\forall x \in \mathbb{R}^n$, sa forme quadratique associée $x^T K x \geq 0$, c'est à dire :

$$\sum_{i,j=1}^{n} x_i x_j K_{i,j} \geq 0$$

. Si la forme quadratique associée à K est nulle uniquement lorsque $x = 0$, alors on dit que K est définie positive.

Remarque. Si $\forall x \in \mathbb{R}^n \setminus \{0\}$, $\sum_{i,j=1}^{n} x_i x_j K_{i,j} > 0$, alors K est définie positive.

Proposition 2.5. *Si une matrice K est définie positive, alors ses valeurs propres sont strictement positives et son déterminant également. Une telle matrice K est donc inversible.*

Ainsi, si notre base $\mathcal{V}$ génère une matrice d'interpolation K définie positive, alors nous pouvons avoir un problème d'interpolation correctement posé. On s'intéresse donc aux fonctions qui pourraient garantir cette propriété pour la matrice d'interpolation.

Définition 2.6. Soit $\phi : \mathbb{R}^d \to \mathbb{R}$ une fonction continue, alors ϕ est dite définie positive si elle est paire et si, $\forall n \in \mathbb{N}^*$, $\forall (x_1, \ldots, x_n) \in \mathbb{R}^d$ distincts deux à deux et $\forall \alpha \in \mathbb{R}^n \setminus \{0\}$, on a :

$$\sum_{i,j=1}^{n} \alpha_i \alpha_j \phi(x_i - x_j) > 0$$

On remarque l'analogie entre le caractère positif définie d'une matrice et d'une fonction. On rappelle que l'on utilise la notation suivante : $\phi(x) = \varphi(\|x\|)$, ce qui nous donne, dans le cadre de la proposition précédente :

$$\sum_{i,j=1}^{n} \alpha_i \alpha_j \varphi(\|x_i - x_j\|) > 0 \iff \sum_{i,j=1}^{n} \alpha_i \alpha_j \phi_{x_i}(x_j) > 0$$

Ainsi, si les fonctions radiales de la base $\mathcal{V}$ sont définies positives, alors la matrice K l'est également. Puisque nos fonctions radiales sont définies par une fonction $\varphi : \mathbb{R} \to \mathbb{R}$ composée avec une norme centrée, on s'intéresse aux propriétés de φ qui induirait le caractère défini positif de la fonction radiale.

2.4.1 Théorème de Bochner

En 1932, Bochner a établi une caractérisation des fonctions définies positives à partir de transformée de fourrier :

Théorème 2.6. *Une fonction $\Phi \in C(\mathbb{R}^d)$ à valeurs complexes est définie positive sur $\mathbb{R}^d$ si et seulement si elle peut être définie comme la transformée de Fourier d'une mesure borélienne μ non-négative sur $\mathbb{R}^d$, c'est à dire*

$$\Phi(x) = \hat{\mu}(x) = \frac{1}{\sqrt{(2\pi)^d}} \int_{\mathbb{R}^d} e^{-i(x,y)} d\mu(y)$$

Preuve. On adapte la définition que l'on a donnée d'une fonction définie positive dans le cas de fonctions à valeurs complexes : Φ est définie positive si, $\forall n \in \mathbb{N}^*, \forall (x_1, \ldots, x_n) \subset \mathbb{R}^d$, distincts deux à deux et $\forall c \in \mathbb{C}^n \setminus \{0\}$,

$$\sum_{j=1}^{n} \sum_{k=1}^{n} c_j \overline{c_k} \Phi(x_j - x_k) \geq 0$$

Ainsi, on considère que Φ est la transformée de Fourier d'une mesure finie μ non-négative. On a alors :

$$\begin{aligned}\sum_{j=1}^{n}\sum_{k=1}^{n} c_j\overline{c_k}\Phi(x_j-x_k) &= \frac{1}{\sqrt{(2\pi)^d}}\sum_{j=1}^{n}\sum_{k=1}^{n}\left[c_j\overline{c_k}\int_{\mathbb{R}^d} e^{-i(x_j-x_k,y)}d\mu(y)\right]\\ &= \frac{1}{\sqrt{(2\pi)^d}}\int_{\mathbb{R}^d}\left(\sum_{j=1}^{n} c_j e^{-i(x_j,y)}\sum_{k=1}^{n}\overline{c_k}e^{i(x_k,y)}\right)\mathrm{d}\mu(y)\\ &= \frac{1}{\sqrt{(2\pi)^d}}\int_{\mathbb{R}^d}\left|\sum_{j=1}^{n} c_j e^{-i(x_j,y)}\right|^2\mathrm{d}\mu(y)\end{aligned}$$

Or, comme μ est une mesure non-négative, on a

$$\frac{1}{\sqrt{(2\pi)^d}}\int_{\mathbb{R}^d}\left|\sum_{j=1}^{n} c_j e^{-i(x_j,y)}\right|^2\mathrm{d}\mu(y)\geq 0.$$

Et donc la fonction Φ est définie positive.

Remarque. On remarque que, puisque la transformée de fourrier d'une gaussienne est une gaussienne, alors la fonction $\Phi(x)=e^{-\alpha\|x\|^2}$, avec $\alpha>0$ est définie positive sur $\mathbb{R}^d$, $\forall d\in\mathbb{N}^*$

Néanmoins, cette caractérisation des fonctions définies positives est utile pour démontrer certains résultats qui suivent, mais n'est pas la plus utile en pratique. Nous préférerons les caractérisations directes de la fonction $\varphi:\mathbb{R}\to\mathbb{R}$ qui, composée avec une norme, nous permettent d'obtenir les fonctions radiales de notre espace d'interpolation.

2.4.2 Fonctions complètement monotones

On s'intéresse maintenant aux fonctions $\varphi:\mathbb{R}\to\mathbb{R}$ et on souhaite caractériser celles qui permettrons à $\phi=\varphi(\|.\|)$ d'être définie positive. On se penche particulièrement sur les fonctions complètement monotones que nous allons chercher à caractériser. Nous verrons ensuite les fonctions à monotonie multiple ainsi que des théorèmes garantissant que les fonctions radiales définies par ces fonctions sont définies positives.

Définition 2.7. Une fonction $\varphi:\mathbb{R}_+\to\mathbb{R}$ telle que $\varphi\in C(\mathbb{R}_+)\cap C^\infty(\mathbb{R}_+^*)$ est dite complètement monotone si

$$\forall k\in\mathbb{N},\forall r>0,\ \ (-1)^k\varphi^{(k)}(r)\geq 0$$

Remarque. D'après la définition, tous les dérivées d'une fonction complètement monotone tendent vers 0 en $+\infty$. En effet, puisqu'elles sont de signes alternés, les dérivées positives seront décroissantes et les dérivées négatives seront croissantes. Comme la définition indique qu'aucune dérivée ne change de signe, alors elles tendent vers 0. Les fonctions complètement monotones sont en fait des fonctions qui tendent relativement vite vers 0.

Exemple. Quelques exemples de fonctions complètement monotones, dont certaines sont tracées dans la figure 5 :

- La fonction constante $x\mapsto c\in\mathbb{R}$, qui sera exclue dans un des théorèmes suivants
- A base d'exponentielle : $x\mapsto e^{-\alpha x}$, $x\mapsto e^{\frac{a}{x}}$, $e^{-\alpha x^\beta}$ avec $\alpha\geq 0$, $0\leq\beta\leq 1$
- A base de logarithme : $\frac{\log(1+x)}{x}$, $\log(b+\frac{c}{x})$ avec $b\geq 1$ et $c>0$
- La fonction rationnelle $x\mapsto\frac{1}{(\lambda+\mu x)^\alpha}$ avec $\lambda,\mu,\nu\geq 0$
- Ou encore des fonctions plus atypiques : $x\mapsto\log\left[\frac{x}{x-\log(1+x)}\right]$, $x\mapsto(1+x)^{\frac{a}{x}}$ avec $a\geq 0$.

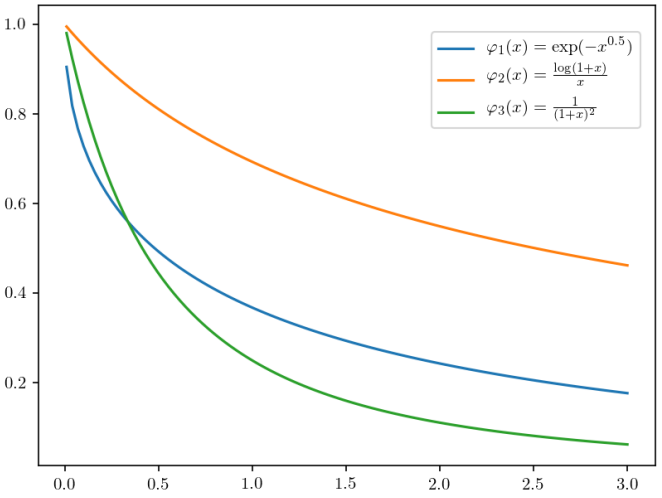


Figure 5: Fonctions complètement monotones dans $\mathbb{R}$

Nous verrons bientôt que chacune de ces fonctions peut alors être utilisées pour définir une fonction radiale définie positive sur leur ensemble de définition en les composant avec une norme comme nous avons vu précédemment $\phi = \varphi(\|.\|^2)$. Cependant, puisque nous aurons besoin de fonctions définies en 0, il nous faudra des fonctions complètement monotones continues en 0, ce qui n'est pas le cas de tous les exemples ci-dessus.

2.4.3 Théorème de Hausdorff-Bernstein-Widder

Le théorème suivant est une caractérisation des fonctions complètement monotones :

Théorème 2.7 (Hausdorff-Bernstein-Widder)**.** *Un fonction $\varphi : \mathbb{R}_+ \to \mathbb{R}$ est complètement monotone sur $\mathbb{R}_+$ si et seulement si c'est la transformée de Laplace d'une mesure de Borel finie non-négative μ sur $\mathbb{R}_+$, c'est à dire que φ est de la forme :*

$$\varphi(x) = \mathcal{L}\mu(x) = \int_0^{+\infty} e^{-xt} \mathrm{d}\mu(t)$$

On note alors $\varphi = \mathcal{L}(d(\mu))$.

Preuve. La preuve peut être trouvée dans [Fas06].

Propositions 2.8. *Les fonctions complètement monotones sont stables pour de nombreuses opérations. Soit $f = \mathcal{L}(d(\mu))$ et $g = \mathcal{L}(d(\nu))$ deux fonctions complètement monotones, on a :*

- $\forall \alpha > 0$, *af est complètement monotone et $af = \mathcal{L}(d(a\mu))$*
- *$f + g$ est complètement monotone et $f + g = \mathcal{L}(d(\mu + \nu))$*
- *fg est complètement monotone et $fg = \mathcal{L}(d(\mu * \nu))$*
- *Si g' est complètement monotone, alors la fonction $x \mapsto e^{-g(x)}$ est complètement monotone*
- *Si $\log(f)$ est complètement monotone, alors f est complètement monotone*
- *Si f est complètement monotone et que g est une fonction positive avec une dérivée complètement monotone, alors $f \circ g$ est complètement monotone*

Preuve. La preuve peut être effectuée en utilisant la caractérisation des fonctions complètement monotones par le théorème 2.6 . La démonstration peut être trouvée dans [Mer14].

Théorème 2.9 (Schoenberg, 1938)**.** *Une fonction φ non constante est complètement monotone sur $\mathbb{R}_+$ si et seulement si $\phi = \varphi(\|.\|^2)$ est définie positive et radiale sur $\mathbb{R}^d$, $\forall d \in \mathbb{N}$.*

Preuve. Le théorème de Hausdorff-Bernstein-Widder implique que l'on peut écrire φ comme

$$\varphi(x) = \int_0^{+\infty} e^{-xt} \mathrm{d}\mu(t)$$

avec μ une mesure de Borel non négative. On peut donc écrire Φ sous la forme :

$$\Phi(x) = \int_0^{+\infty} e^{-\|x\|^2 t} \mathrm{d}\mu(t)$$

Ainsi, en reprenant la définition d'une fonction définie positive, on veut montrer que, $\forall c \in \mathbb{R}^n$:

$$\sum_{i,j=1}^{n} c_i c_j \Phi(x_i - x_j) = \int_0^{+\infty} \sum_{i,j=1}^{n} c_i c_j e^{-\|x_i - x_j\|^2 t} \mathrm{d}\mu(t)$$

Or, la fonction gaussienne étant elle-même définie positive comme nous avons vu dans la partie 2.4.1. De plus, puisque φ n'est pas constante, cela implique que la mesure μ n'est pas concentrée sur l'origine, ce qui permet, avec le caractère défini positif de la fonction gaussienne de conclure :

$$\int_0^{+\infty} \sum_{i,j=1}^{n} c_i c_j e^{-\|x_i - x_j\|^2 t} \mathrm{d}\mu(t) > 0$$

Remarque. Cette caractérisation des fonctions définies positives est indépendante de la dimension d de l'espace de définition de la fonction. Nous verrons ensuite la définition des fonctions à monicité multiple pour lesquelles la propriété dépendra de la dimension de l'espace $\mathbb{R}^d$.

Grâce à ce théorème, nous avons une première caractérisation de fonctions nous permettant d'assurer l'inversibilité de notre matrice d'interpolation et donc de pouvoir bien poser le problème d'interpolation. On rappelle que, jusque maintenant, nous avons imposé les conditions suivantes dans notre problème d'interpolation : $\dim(\mathcal{V}) = \mathrm{card}(\mathcal{X})$, $\forall \phi \in \mathcal{V}, \exists x \in \mathcal{X}$ tel que $\phi = \varphi(\|. - x\|)$ et dorénavant le caractère positif défini de la fonction ϕ qui revient à ce que φ soit complètement monotone.

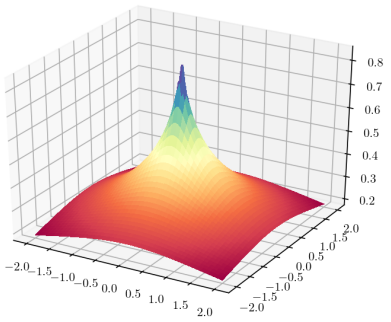
Voici quelques exemples de fonctions ϕ radiales définies positives en utilisant les exemples de fonctions φ complètement monotones précédemment décrites :

2.4.4 Fonctions à monotonicité mutliple

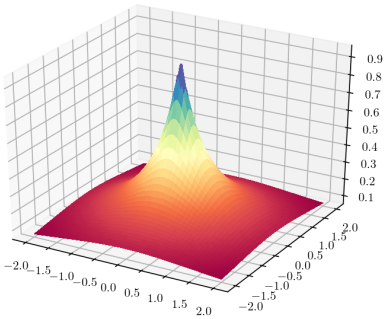
Nous disposons dores et déjà d'une classe de fonction radiales définies positives induites par des fonctions complètement monotones. Néanmoins, il existe une classe de fonction plus générale qui permet également de garantir que les fonctions radiales induites par ces dernières sont définies positives : il s'agit des fonctions à monotonicité multiple.

Définition 2.8. Un fonction réelle $\varphi \in C^{s-2}(\mathbb{R}_=^*)$ où $s \geq 2$ telle que $(-1)^k \varphi^{(k)}(x)$ est positive, décroissante et convexe pour $0 \leq k \leq s-2$ est dite monotone de multiplicité s sur $\mathbb{R}_+^*$. Dans le cas ou $s = 1$, on requiert uniquement que φ soit continue sur $\mathbb{R}_+^*$, positive et décroissante.

Exemple. Soit $\alpha \in \mathbb{N}^*$, la fonction $\varphi : x \mapsto (1-x)_+^\alpha$ est monotone de multiplicité α.

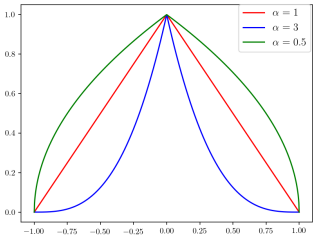


(a) $\varphi_1(x) = \exp(-\|x\|^{0.5})$

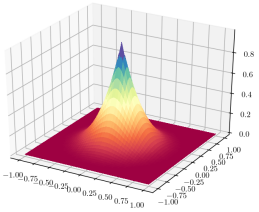


(b) $\varphi_3(x) = \frac{1}{(1+\|x\|)^2}$

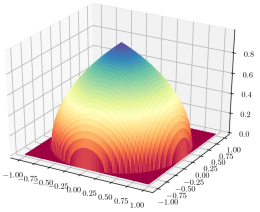
Figure 6: Fonctions complètement monotones dans $\mathbb{R}^2$



(a) $\varphi(|x|)$ pour $\alpha = 0.5, 1, 3$



(b) $\varphi(\|x\|)$ avec $\alpha = 3$



(c) Fonction $\varphi(\|x\|)$ avec $\alpha = 3$

Figure 7: φ dans $\mathbb{R}$ et $\mathbb{R}^2$ pour différentes valeurs de α

Théorème 2.10. *Un fonction continue $\varphi : \mathbb{R}_+^* \to \mathbb{R}$ est monotone de multiplcité $k \in \mathbb{N}^*$ sur $\mathbb{R}_+^*$ si et seulement si elle est de la forme*

$$\varphi(x) = \int_0^{+\infty} (1 - xt)_+^{k-1} \mathrm{d}\mu(t)$$

avec μ une mesure borélienne positive sur $\mathbb{R}_+^*$.

Théorème 2.11 (Micchelli's paper et Buhman)**.** *Soit $k = \lfloor d/2 \rfloor + 2$ un entier naturel. Si $\varphi : \mathbb{R}_+^* \to \mathbb{R}$ est monotone de multiplicité k sur $\mathbb{R}_+^*$ et non constante, alors $\Phi = \varphi(\|.\|^2)$ est définie positive et radiale sur $\mathbb{R}^d$.*

Finalement, on dispose dorénavant d'une boîte à outils théorique relativement complète afin de construire nos espaces d'interpolation. Nous utiliserons l'ensemble de ces résultats dans chacune des parties suivantes. On note quand même qu'il aurait pu être intéressant de développer également la notion de fonction conditionnellement définie positive, telle qu'elle est abordée dans [Mar18] par exemple.

3 Interpolation à base de fonctions radiales

On reprend l'ensemble des notations décrites précédemment. On suppose qu'on dispose d'un ensemble de points $\mathcal{X} = (x_1, \ldots, x_n) \subset \mathbb{R}^d$ et $\mathcal{Y} \subset \mathbb{R}$ qui correspondent aux mesures de la fonction f qu'on cherche à approcher dans une base de fonctions radiales $\mathcal{V} = (\phi_{x_1}, \ldots, \phi_{x_n})$ de même taille n que notre échantillon de points. Chaque fonction ϕ de $\mathcal{V}$ est définie par une fonction associée φ telle que $\forall x \in \mathcal{X},\ \phi_x = \varphi(\|. - x\|)$. Dans la partie précédente, nous avons montré que, pour que le problème soit bien posé, nous avions besoin que les fonctions radiales de la base d'approximation $\mathcal{V}$ soient définies positives, ce qui était garanti dans le cas où les fonctions φ sont complètement monotones (ou à monotonicité multiple d'ordre k avec des contraintes liants k et d).
L'interpolation Φ de f dans la base $\mathcal{V}$, notée Φ_f est donc de la forme

$$\Phi_f(x) = \sum_{i=1}^{n} \alpha_i \phi_{x_i}(x) = \sum_{i=1}^{n} \alpha_i \varphi(\|x - x_i\|)$$

avec les coefficients $(\alpha_i)_{0 \leq i \leq n}$ déterminés grâce à la résolution du système linéaire associé au problème d'interpolation. Afin d'évaluer la qualité d'un interpolant, on introduit une mesure de l'erreur entre ce dernier et la fonction qu'il cherche à approcher :

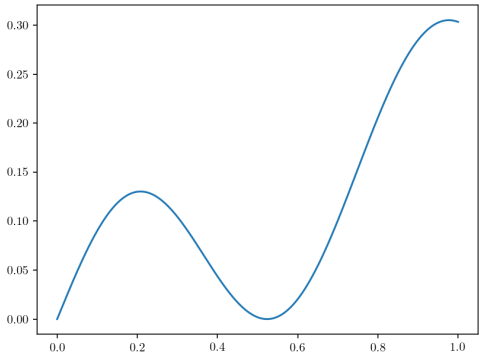
Définition 3.1. Soif $f : \mathbb{R}^d \to \mathbb{R}$ et Φ_f son interpolant dans une base de fonctions radiales $\mathcal{V}$. On définit l'écart quadratique moyen sur un ensemble de points $(x_1, \ldots, x_n)$ entre f et Φ_f tel que :

$$\mathrm{EQM}(\Phi_f) = \frac{1}{\sqrt{n}} \|\Phi_f - f\|_2 = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (\Phi_f(x_i) - f(x_i))^2}$$

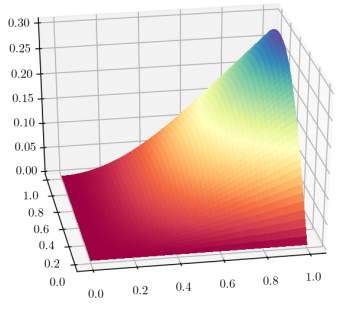
3.1 Interpolation d'une fonction f connue

Avant de discuter de la paramétrisation de l'interpolation afin d'améliorer sa précision, nous voulons donner un premier aperçu de résultats obtenus sur des exemples simples. Dans l'ensemble de cette partie, on considérera 2 fonctions que nous chercherons à approximer :

- $f_1 : [0, 1] \to \mathbb{R}$, telle que $f_1(x) = \sin x \cos(3x)^2 \exp^{-x^2}$
- $f_2 : [0, 1]^2 \to \mathbb{R}$, telle que $f_2(x, y) = y \sin(x)^2 \exp^{-y^2}$



(a) $f_1 : x \mapsto \sin x \cos(3x)^2 \exp^{-x^2}$

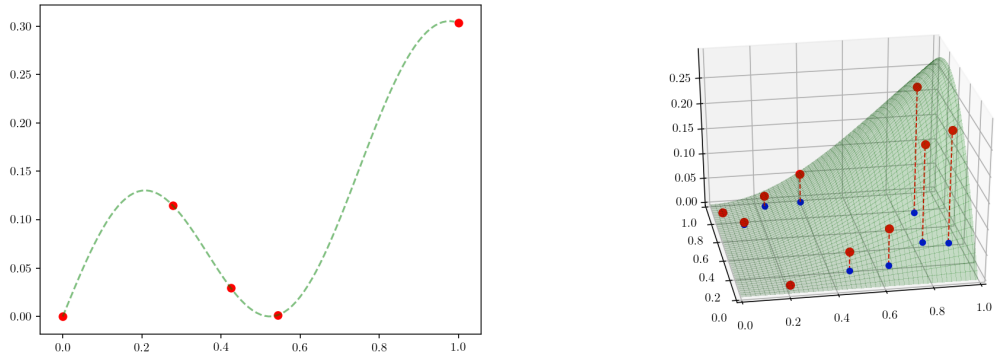


(b) $f_2 : (x, y) \mapsto y \sin(x)^2 \exp^{-y^2}$

Figure 8: Deux exemples de fonctions à interpoler

3.1.1 Première approche

Ainsi, nous construirons nos ensembles d'interpolation $(\mathcal{X}, \mathcal{Y})$ en tirant un ensemble de points aléatoires dans le domaine de chaque fonction. On en sélectionne 5 pour f_1 et 10 pour f_2 et évaluons la fonction f à chacun de ces points, ce qui nous donne les discrétisations suivantes de la Figure 9. Dans le cas de l'interpolation de f_1, nous imposerons les bords du domaine $\{0, 1\}$ dans les points d'interpolation afin de permettre une comparaison cohérente avec des interpolations polynomiales.



(a) Discrétisation de f_1 (b) Discrétisation de f_2

Figure 9: Discrétisations des fonctions dans $\mathbb{R}$ et $\mathbb{R}^2$

On dispose donc dorénavant de $(\mathcal{X}_1, \mathcal{Y}_1)$ et $(\mathcal{X}_2, \mathcal{Y}_2)$ et on souhaite interpoler une solution à l'aide de fonction radiales. On sélectionne donc une fonction complètement monotone $\varphi(x) = \exp(-\epsilon^2 x)$, avec $\epsilon > 0$ le paramètre de forme, et on construit la base de fonctions radiales définies positives centrées en chaque point de $\mathcal{X}$: $\mathcal{V} = \text{vect}\left\{\phi_c(x) = \varphi(\|x - c\|^2),\ c \in \mathcal{X}\right\}$. Notre solution dans cette base est donc de la forme $\Phi = \sum_{i=1}^{n} \alpha_i \phi_i$ avec $n = \dim(\mathcal{V}) = \text{card}(\mathcal{X})$, ce qui nous amène à résoudre le système linéaire suivant (dans le cas de f_1) pour trouver le vecteur des poids $\alpha \in \mathbb{R}^n$:

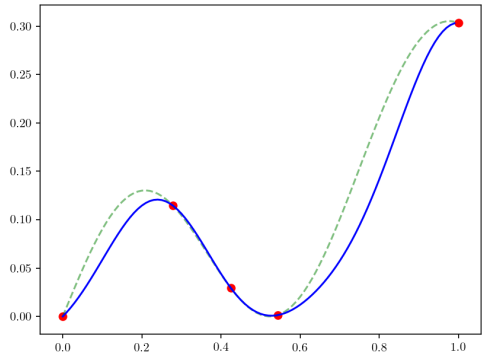
$$\begin{pmatrix} \varphi(0) & \varphi(\|x_0 - x_1\|^2) & \varphi(\|x_0 - x_2\|^2) & \varphi(\|x_0 - x_3\|^2) \\ \varphi(\|x_1 - x_0\|^2) & \varphi(0) & \varphi(\|x_1 - x_2\|^2) & \varphi(\|x_1 - x_3\|^2) \\ \varphi(\|x_2 - x_0\|^2) & \varphi(\|x_2 - x_1\|^2) & \varphi(0) & \varphi(\|x_2 - x_3\|^2) \\ \varphi(\|x_3 - x_0\|^2) & \varphi(\|x_3 - x_1\|^2) & \varphi(\|x_3 - x_2\|^2) & \varphi(0) \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} = \begin{pmatrix} f_1(x_0) \\ f_1(x_1) \\ f_1(x_2) \\ f_1(x_3) \end{pmatrix}$$

En procédant de façon similaire pour f_2, on obtient les interpolations de la Figure 10.

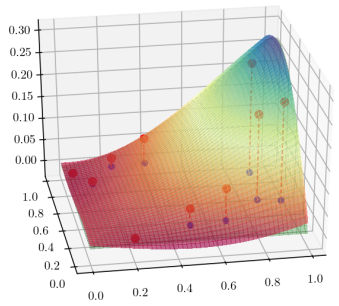
Il est assez intéressant de se pencher sur la décomposition de f_1 dans la base de fonctions radiales. A partir de la solution du système linéaire décrite dans la Table 1, on peut tracer chacune des fonctions $(\alpha_i \phi_i)_{0 \leq i \leq 4}$ dont la somme est la solution de notre problème d'interpolation. Pour introduire l'influence du paramètre de forme dans l'interpolation, nous décomposerons la solution pour deux paramètres de formes : $\epsilon = 4.5$ comme précédemment, puis $\epsilon = 1$ dans la Figure 11.

i =	0	1	2	3	4
Valeur de x_i	0	0.27	0.42	0.54	1
Valeur de α_i $(\epsilon = 4.5)$	-0.038	0.201	-0.142	0.055	0.303
Valeur de α_i $(\epsilon = 1)$	-199.81	1311.57	-2249.60	1188.95	-55.19

Table 1: Solution du système linéaire pour f_1

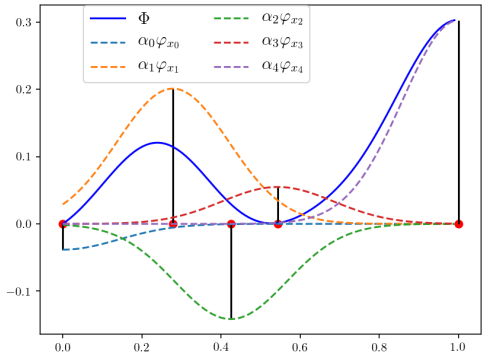


(a) EQM(Φ_{f_1}) = 0.1302, $\epsilon = 4.5$

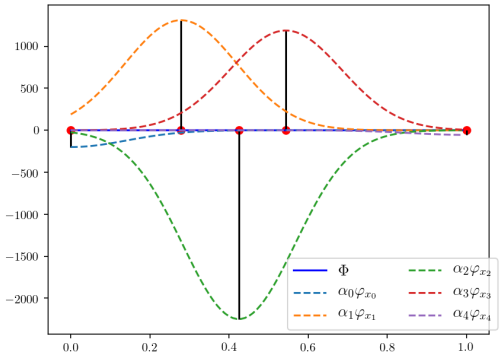


(b) EQM(Φ_{f_2}) = 0.167, $\epsilon = 0.8$

Figure 10: Interpolation de f_1 et de f_2 à l'aide de fonctions radiales gaussiennes



(a) $\epsilon = 4.5$, EQM= 0.1302



(b) $\epsilon = 1$, EQM=0.116

Figure 11: Décomposition de Φ_{f_1} pour deux paramètres de forme ϵ différents

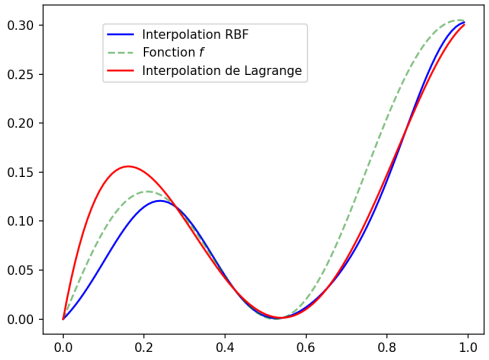
On remarque déjà que, pour un coefficient de forme plus petit, nous avons une plus grande précision dans l'interpolation (EQM plus petit) mais nos coefficients ont tendance à exploser et prendre des valeurs anormalement élevées au vu des valeurs réelles de la fonction f_1, ce qui nuit au conditionnement de la matrice d'interpolation.

Dans la Figure 12 nous comparons notre interpolation avec d'autres méthodes classiques d'interpolation et l'on observe que l'interpolation RBF obtient un meilleur EQM que les autres méthodes d'interpolation sur cet exemple-là.

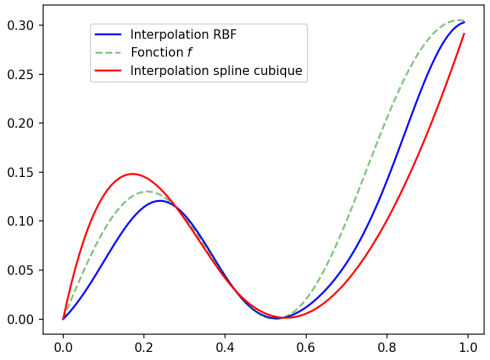
3.1.2 Le choix de la fonction φ

Comme nous avons pu le voir précédemment, il y a de nombreux choix de fonctions φ telles que $\phi(x) = \varphi(\|x\|)$ soient radiales définies positives pour construire notre espace d'interpolation. Nous avons décidé d'en retenir quelques une seulement, que nous avons résumé dans le Tableau 2, dans l'optique de comparer leurs performances pour interpoler les fonctions f_1 et f_2.

Afin de se faire une idée des fonctions radiales engendrées par ces fonctions, et donc de l'espace d'interpolation, la Figure 13 illustre ces fonctions dans $\mathbb{R}$ et la Figure 14 montre certaines d'entre elles dans $\mathbb{R}^2$.



(a) Interpolation de Lagrange (EQM = 0.1391)



(b) Interpolation spline cubique (EQM = 0.2263)

Figure 12: Comparaison entre RBF et d'autres méthodes d'interpolation

Fonction φ	Abbréviation	Définition
Gaussienne	GA	$\varphi(x) = e^{-(\epsilon x)^2}$
Fonction tronquée	FT	$\varphi(x) = (1-x)_+^\epsilon$
Multiquadratique Inverse	MQI	$\varphi(x) = \frac{1}{\sqrt{1+(\epsilon x)^2}}$
C^0 Matern	M0	$\varphi(x) = e^{-\epsilon x}$
C^2 Thin Plate Splin	TPS	$\varphi(x) = e^{-\epsilon x}(1+\epsilon x)$
C^4 Matern	M4	$\varphi(x) = e^{-\epsilon x}.(3+3\epsilon x+(\epsilon x)^2)$

Table 2: Exemples de fonctions associées à des fonctions radiales définies positives

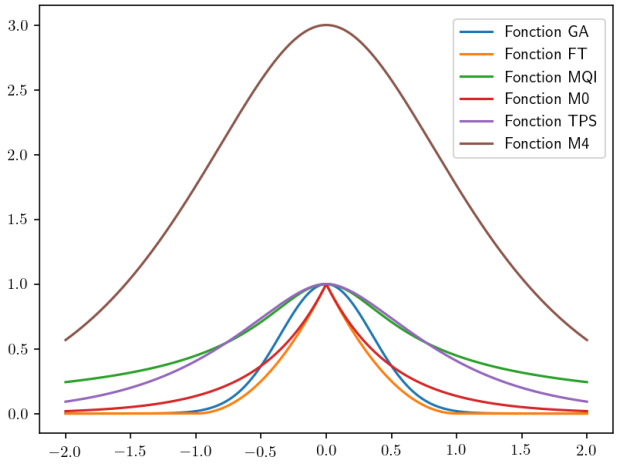
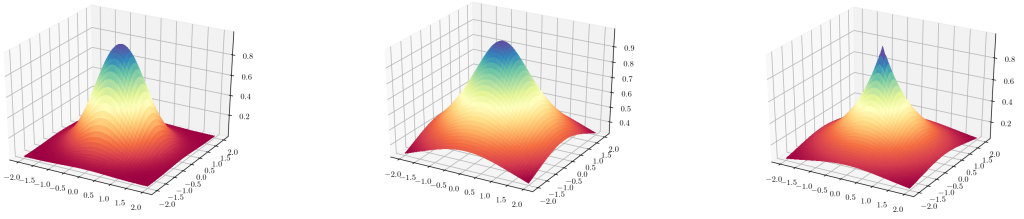


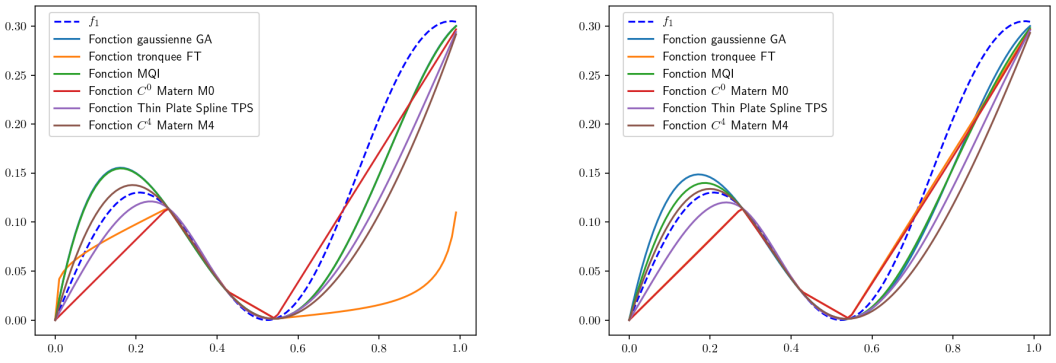
Figure 13: Courbes des fonctions radiales dans $\mathbb{R}$ avec $\epsilon = 2$

On teste chacune d'entre elles pour l'interpolation de f_1 avec les plusieurs valeurs de ϵ et on reporte les différentes interpolation dans la Figure 15 et on reporte dans le Tableau 3 les EQM correspondants à chaque interpolation. De ces exemples sur f_1 on peut voir que chaque fonction φ approche relativement différemment f_1 et que l'influence du paramètre de forme ϵ sur la forme de la solution interpolée n'est pas la même du tout. De plus, il n'y a pas une fonction qui serait plus efficace quelque soit le paramètre de forme utilisé, comme on le voit dans le Tableau 3.

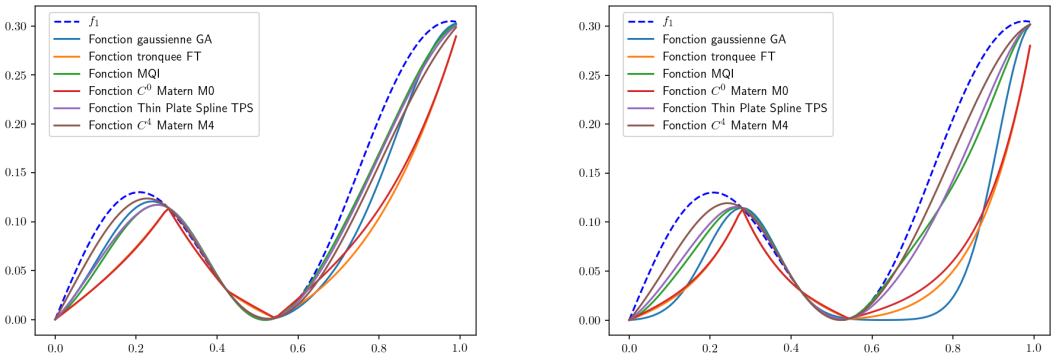


(a) GA : $\phi(x) = e^{-(\epsilon\|x\|)^2}$ (b) MQI : $\phi(x) = \frac{1}{\sqrt{1+(\epsilon\|x\|)^2}}$ (c) M0 : $\phi(x) = e^{-\epsilon\|x\|}$

Figure 14: Surfaces de fonctions radiales dans $\mathbb{R}^2$ avec $\epsilon = 1$



(a) $\epsilon = 0.2$ (b) $\epsilon = 1$



(c) $\epsilon = 4.5$ (d) $\epsilon = 8$

Figure 15: Comparaison des interpolations en fonction de φ et de ϵ

Fonction φ	GA	FT	MQI	M0	TPS	M4
$\epsilon = 0.2$	0.138	0.516	0.137	**0.13**	0.175	0.207
$\epsilon = 1$	0.116	0.13	**0.11**	0.135	0.149	0.185
$\epsilon = 4.5$	0.13	0.242	**0.096**	0.227	0.097	0.101
$\epsilon = 8$	0.368	0.363	0.159	0. 344	0.143	**0.084**

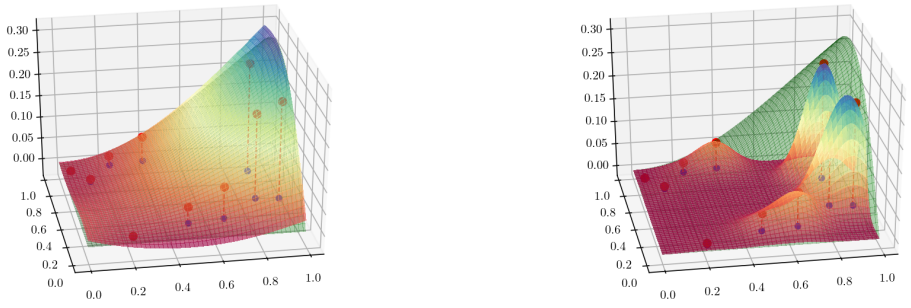
Table 3: EQM(Φ_{f_1}) en fonction de φ et de ϵ (en gras : minimum par ligne)

Dans le cas de l'interpolation de f_2, on obtient les résultats résumés dans le Tableau 4.

Fonction φ	GA	FT	MQI	M0	TPS	M4
$\epsilon = 0.2$	0.293	7.163	0.275	0.695	0.472	**0.218**
$\epsilon = 1$	**0.171**	0.621	0.312	0.649	0.496	0.251
$\epsilon = 4.5$	1.327	1.398	0.623	1.184	0.529	**0.478**
$\epsilon = 8$	2.578	2.288	0.901	2.071	0.903	**0.597**

Table 4: EQM(Φ_{f_2}) en fonction de φ et de ϵ (en gras : minimum par ligne)

On observe des conclusions similaires quant aux résultats de l'interpolation de f_2. Globalement, la fonction φ M4 est plus précise mais l'EQM minimum dans le Tableau 4 est atteint avec une fonction Gaussienne et un paramètre de forme $\epsilon = 1$. On représente dans la Figure 16 la meilleure interpolation M4 et la pire interpolation Gaussienne.



(a) M4 avec $\epsilon = 0.2$ (b) GA avec $\epsilon = 8$

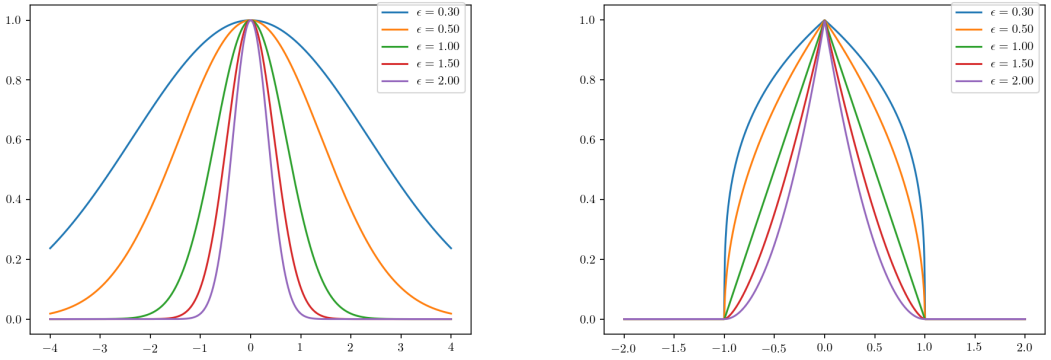
Figure 16: Deux interpolations de f_2

Dans la figure (b), on observe que le paramètre de forme élevé amène les fonctions gaussiennes à avoir un support très compact autour de leur centre, ce qui limite essentiellement leur impact au fait "d'aller chercher" la valeur de f_2 au point d'interpolation.

3.1.3 Le choix du paramètre de forme ϵ

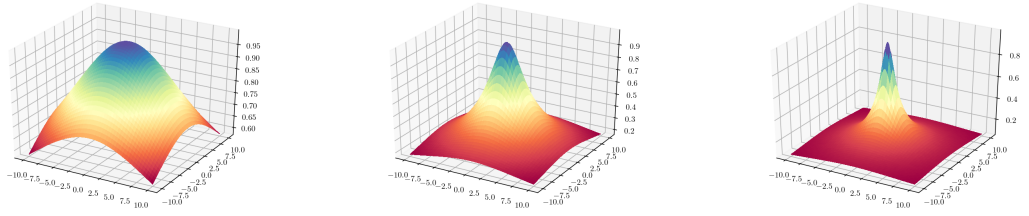
Comme nous avons pu le constater dores et déjà précédemment, le paramètre de forme ϵ, ou coefficient d'aplatissement, influence la forme des fonctions radiales de notre espace d'interpolation $\mathcal{V}$ et la précision de la solution interpolée. On propose quelques illustrations de l'impact du paramètre de forme sur les fonctions radiales dans la Figure 17 et dans la Figure 18.

D'une manière générale, un paramètre élevé réduira le support de la fonction radiale (i.e. l'ensemble ou elle prend des valeurs non-nulles) et donc atténuera son impact sur l'interpolant global. C'est légèrement différent dans le cas de la fonction tronquée FT, puisque son support est fixé égal à la boule de rayon 1, où le paramètre de forme influe sur le caractère convexe ou concave de la fonction. Nous nous intéressons principalement à son influence sur la qualité de l'interpolation. On est donc intéressés par les courbes décrivant la valeur de l'Ecart Quadratique Moyen en fonction de la valeur d'ϵ utilisée pour l'interpolation dans la Figure 19.



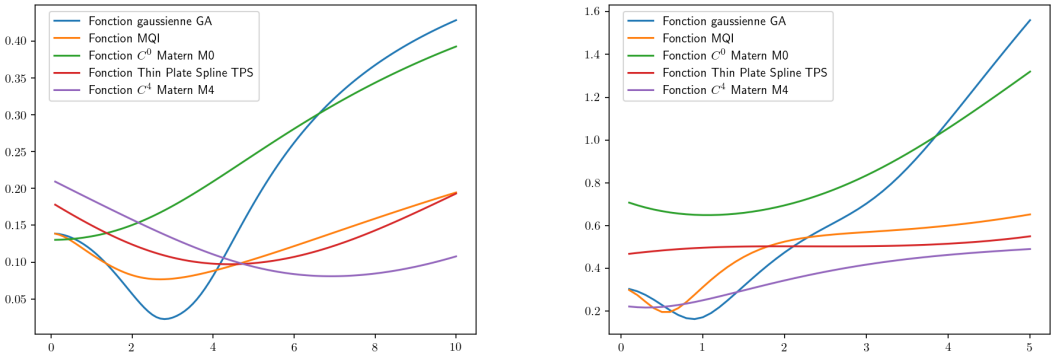
(a) Fonction gaussienne GA (b) Fonction tronquée FT

Figure 17: Influence du paramètre de forme sur les fonctions radiales dans $\mathbb{R}$



(a) MQI avec $\epsilon = 0.1$ (b) MQI avec $\epsilon = 0.4$ (c) MQI avec $\epsilon = 1$

Figure 18: Influence du paramètre de forme sur la fonction MQI dans $\mathbb{R}^2$



(a) EQM(Φ_{f_1}) en fonction de la valeur de ϵ (b) EQM(Φ_{f_2}) en fonction de la valeur de ϵ

Figure 19: Ecarts quadratiques moyens pour f_1 et f_2 en fonction de ϵ

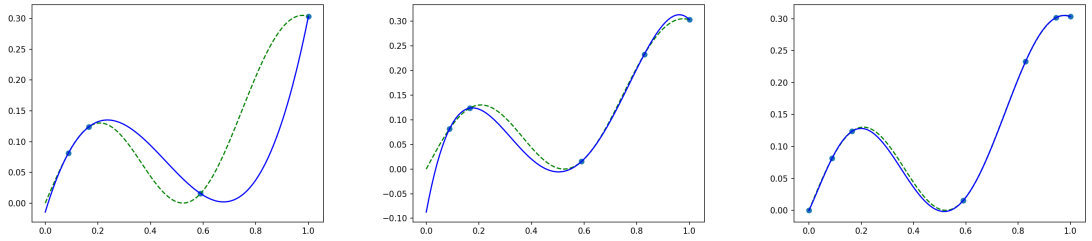
Dans ce cadre là, il apparaît que la recherche du paramètre de forme optimal se réduit à la recherche du minimum de la fonction $\epsilon \mapsto \text{EQM}(\Phi_f)$. Sauf que nous pouvons actuellement évaluer l'EQM sur un maillage uniforme puisque l'on dispose de l'expression de la fonction f en tout point. Dans des conditions réelles d'interpolations, on n'a pas l'expression de f mais uniquement ses valeurs en les points de $\mathcal{X}$. Il peut alors être utile de se servir des méthodes de validations croisées en séparant $\mathcal{X}$ entre un ensemble d'interpolation et un ensemble de test pour contrôler la qualité de l'interpolation.

3.2 Optimisation de l'interpolation

3.2.1 Sélection d'un sous ensemble de $\mathcal{X}$

Il existe de nombreux cas dans lesquels on peut se retrouver avec de très nombreux points mesurés dans $(\mathcal{X}, \mathcal{Y})$. Même si réaliser l'interpolation sur l'ensemble des points peut permettre une meilleure précision, cela peut rapidement être trop couteux en termes de calcul numérique et ne pas apporter un supplément de précision justifiant cette complexité.
On peut alors être intéressés par des algorithmes permettant de sélectionner des sous-ensembles de $\mathcal{X}$ afin de garantir une certaine précision avec un minimum de points. On part donc d'un point tiré aléatoirement dans $\mathcal{X}$ et on cherche à rajouter le point pour lequel notre interpolation est la plus mauvaise à chaque itération, jusqu'à atteindre une précision donnée ou un certain nombre de points.

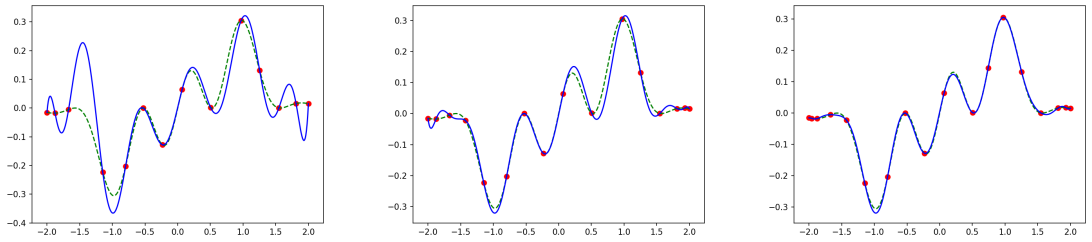
On reprend la fonction f de la partie précédente et on génère un échantillon de 100 points uniformément répartis sur $[0, 1]$. Les résultats de l'algorithme pour différentes exigences de précisions sont représentés dans la Figure 20.



(a) Exigée : 0.8, obtenue : 0.77 (b) Exigée : 0.2, obtenue : 0.2 (c) Exigée : 0.05, obtenue : 0.04

Figure 20: Algorithme de sélection d'un sous ensemble de $\mathcal{X}$ avec différents critères de précision exigés (en rouge les points sélectionnés par l'algorithme)

On reproduit la même expérience sur l'intervalle $[-2, 2]$ sur lequel la fonction f propose beaucoup plus de variations et on présente les résultats obtenus dans la Figure 21. On remarque assez aisément que, dans les graphiques (b) et (c), les points d'interpolation se sont placés naturellement dans la plupart des lieux ou la dérivée de f_1 s'annule, afin de capturer le maximum d'informations sur les variations de la fonction.

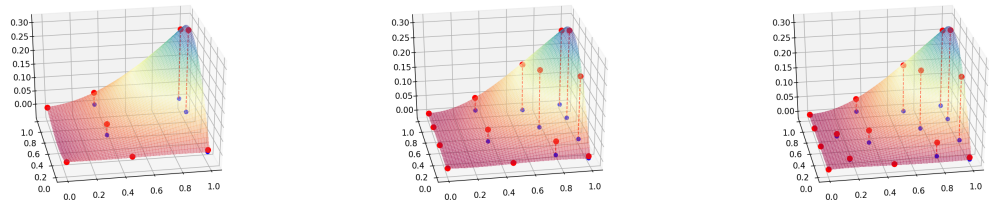


(a) Exigée : 1.5, obtenue : 1.42 (b) Exigée : 0.5, obtenue : 0.39 (c) Exigée : 0.1, obtenue : 0.10

Figure 21: Algorithme de sélection d'un sous ensemble de $\mathcal{X}$ avec différents critères de précision exigés (en rouge les points sélectionnés par l'algorithme)

On reproduit la même expérience sur $[0, 1]^2$ avec la fonction g en utilisant un échantillon de 200 points aléatoires et on représente les résultats obtenus dans la Figure 23.

Ce type d'algorithme peut être particulièrement efficace lors de l'interpolation en conditions réelles lorsqu'on ne dispose pas de l'expression exacte de la fonction et que l'ensemble des points mesurés est de cardinal très élevé. Réaliser l'interpolation sur l'ensemble des points serait trop couteux numériquement et on devrait pouvoir obtenir de très bons résultats en sélectionnant juste



(a) Exigée : 1.5, obtenue : 1.36 (b) Exigée : 0.5, obtenue : 0.47 (c) Exigée : 0.1, obtenue : 0.09

Figure 22: Algorithme de sélection d'un sous ensemble de $\mathcal{X}$ avec différents critères de précision exigés (en rouge les points sélectionnés par l'algorithme)

un sous-ensemble de points qui décrivent au mieux les variations de la fonction qu'on cherche à interpoler.

3.2.2 Ajout d'un terme polynomial

De nombreuses raisons peuvent amener à rajouter un terme polynomial dans l'interpolation à base de fonctions radiales :

- Cela peut améliorer le conditionnement de la matrice d'interpolation
- L'expression polynomiale permet de capturer la tendance de la fonction lorsque les termes radiaux permettent d'ajuster localement aux points d'interpolation
- Lorsqu'on utilise des fonctions qui ne sont pas définies positives, cela peut permettre à la matrice d'être inversible

Comme la méthode d'interpolation à base de fonctions radiales peut engendrer des problèmes de stabilité numérique provenant du mauvais conditionnement de la matrice d'interpolation, l'ajout d'un terme polynomial peut améliorer le conditionnement de la matrice.

Soit $\Pi_m(\mathbb{R}^d)$ l'espace vectoriel des polynômes de degré au plus m de $\mathbb{R}^d$ dans $\mathbb{R}$, et $\mathcal{P} = (p_0, \dots, p_m)$ une base de cette espace. Notre espace d'approximation $\mathcal{V}$ s'écrit alors vect$(\phi_1, \cdots, \phi_n, p_0, \cdots, p_m)$ et la solution Φ au problème d'interpolation est de la forme :

$$\Phi(x) = \sum_{k=0}^{n} \alpha_k \phi_{x_k}(x) + \sum_{j=0}^{m} \gamma_j p_j(x)$$

On suppose de plus que :

$$\sum_{i=0}^{n} \alpha_i p_j(x_i) = 0 \; \forall j \in [\![0, m]\!]$$

ce qui nous permet d'écrire le problème d'interpolation avec le système matriciel suivant :

$$\begin{pmatrix} K & P \\ P^T & O \end{pmatrix} \begin{pmatrix} \alpha \\ \gamma \end{pmatrix} = \begin{pmatrix} Y \\ 0 \end{pmatrix}$$

où $P \in \mathcal{M}_{n,m}$ est telle que $P_{i,j} = p_j(x_i)$. Le système est alors équivalent à :

$$\begin{cases} K\alpha + P\gamma = y \\ P^T\alpha = 0 \end{cases}$$

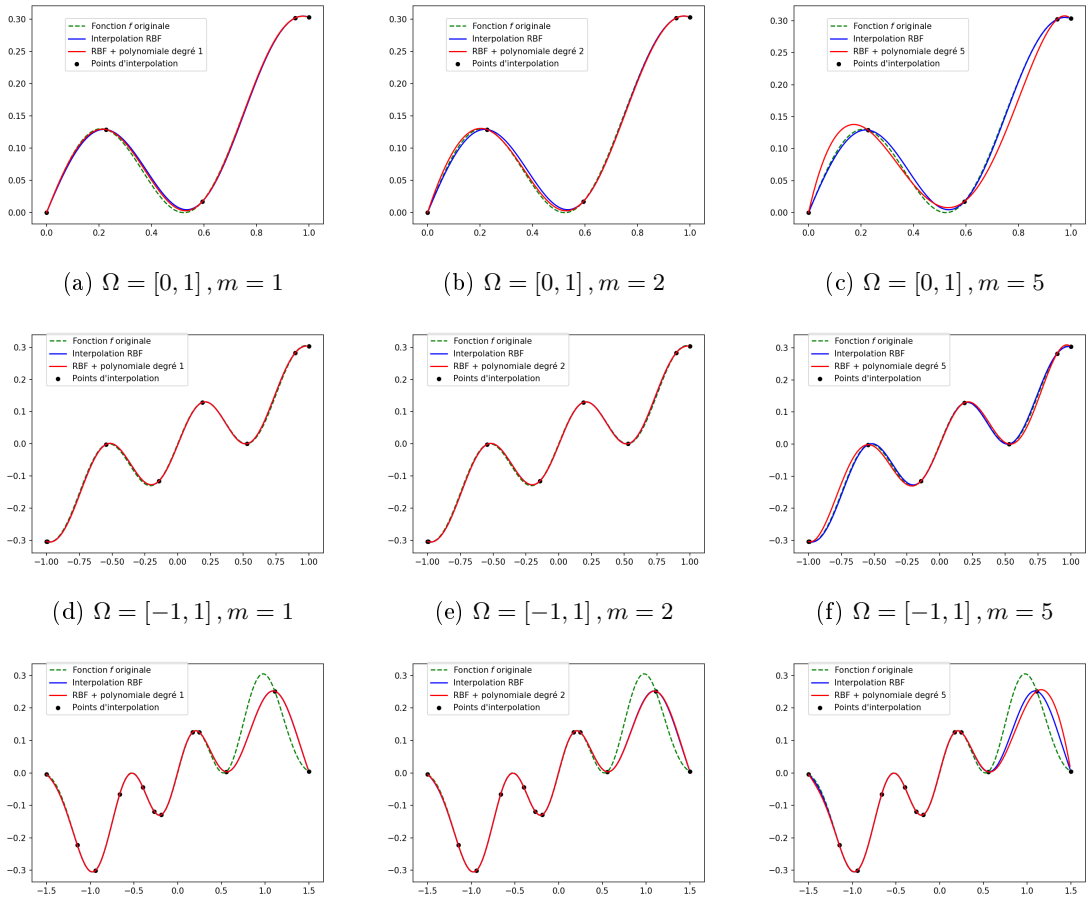
L'addition du terme polynômial garantit par ailleurs une précision d'ordre m à notre interpolant, ce qui signifie que si nos données $(\mathcal{X}, \mathcal{Y})$ sont issues d'un polynôme de degré inférieur à m, alors

notre interpolant Φ sera égal à ce polynôme.

On réalise maintenant l'interpolation à l'aide du terme polynomial sur la fonction f_1 précédemment utilisée sur différents intervalles de $\mathbb{R}$ et pour différents degrés du membre polynomial dans la Figure 23. Nous réalisons autant de tests car il s'avère que l'ajout du terme polynomial n'est bénéfique que dans certaines conditions. On résume l'ensemble des résultats dans le Tableau 5.

Intervalle Ω	Type de résultat	RBF seule	$m = 1$	$m = 2$	$m = 5$
$\Omega = [0, 1]$	EQM	0.02	**0.013**	0.0135	0.068
Nombre de points : 5	Cond(K)	4199	4778	**3188**	4199
$\Omega = [-1, 1]$	EQM	**0.018**	0.020	0.019	0.055
Nombre de points : 8	Cond(K)	325309	353919	**300347**	35362
$\Omega = [-1.5, 1.5]$	EQM	**0.18**	0.182	0.186	0.25
Nombre de points : 12	Cond(K)	**137895**	161364	170493	231048

Table 5: Comparaison des EQM et du conditionnement de la matrice



(a) $\Omega = [0, 1], m = 1$ (b) $\Omega = [0, 1], m = 2$ (c) $\Omega = [0, 1], m = 5$

(d) $\Omega = [-1, 1], m = 1$ (e) $\Omega = [-1, 1], m = 2$ (f) $\Omega = [-1, 1], m = 5$

(g) $\Omega = [-1.5, 1.5], m = 1$ (h) $\Omega = [-1.5, 1.5], m = 2$ (i) $\Omega = [-1.5, 1.5], m = 5$

Figure 23: Comparaison entre interpolation RBF et ajout de terme polynomial de différents degrés sur différents intervalles

On observe dans le Tableau 5 que le terme polynomial permet une bien meilleure précision (EQM plus faible) dans le cas où $\Omega = [0, 1]$. Mais surtout, il permet bien d'avoir un meilleur conditionnalement de la matrice d'interpolation dans les deux premiers cas, ce qui mériterait d'être approfondi afin de garantir une meilleur stabilité numérique de l'interpolation.

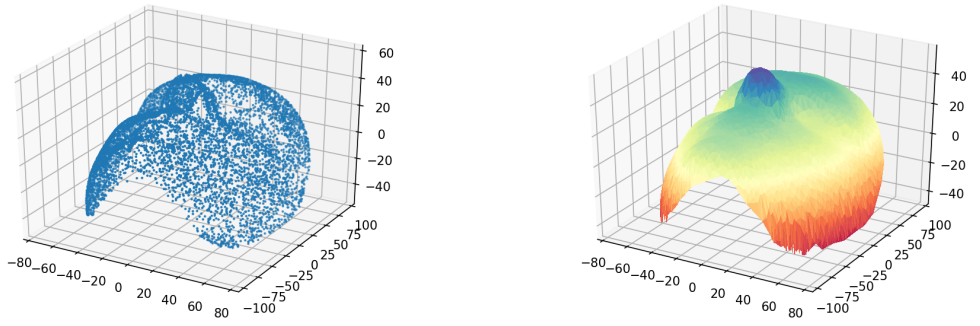
3.3 Interpolation à partir d'un échantillon

On s'intéresse maintenant à l'interpolation de fonctions beaucoup plus complexes dont on ne dispose que d'échantillons de valeurs pour un certain nombre de points. Dans chacun des cas, nous disposons d'un grand nombre de points car les données sont issues de fichiers servant à produire des maquettes 3D, mais nous sommes intéressés par le fait de reconstruire les objets avec le moins de points possibles. Les deux exemples qui suivent, bien que différents, sont inspirés de [dT08].

3.3.1 Reconstruction d'un visage

Dans le premier cas, on s'intéresse à l'interpolation d'un visage d'homme quelconque (*average male face*) décrit par 394 134 points dans $\mathbb{R}^2$. On trace dans la Figure 24 un sous ensemble de l'échantillons de points et une reconstruction à l'aide de triangles afin de bien faire apparaître la forme du visage.

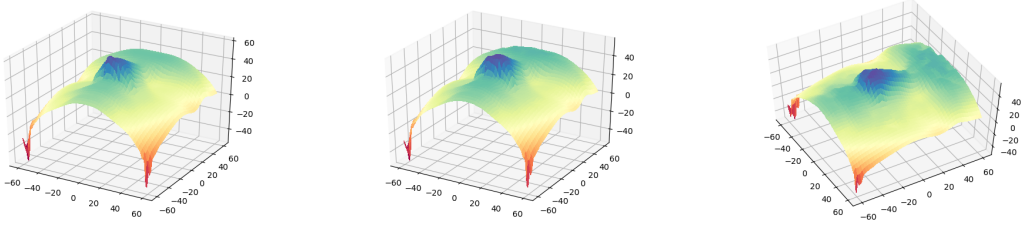
On applique ensuite notre algorithme d'interpolation à base de fonctions radiales sur un ensemble



(a) Echantillon de 7882 points (b) Surfaces triangulaires sur 3941 points

Figure 24: Echantillon de points décrivant un visage humain

réduit de 2628 points, soit 0.6% du nombre de points originaux. On utilise une fonction Thin Plate Spline et plusieurs paramètres de forme et on représente les résultats obtenus dans la Figure 25. Nous avons coupé les bords du visage car les trop grandes variations sur le bord amenait l'interpolant à prendre des valeurs extrêmes dans les coins du maillage.



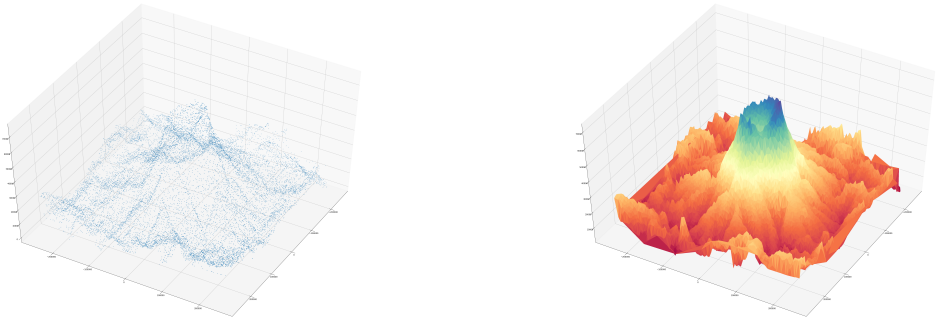
(a) $\epsilon = 0.1$ (b) $\epsilon = 0.3$ (c) $\epsilon = 0.4$

Figure 25: Interpolation sur un échantillon de 2628 points décrivant un visage humain

Au vu de la précision des résultats avec uniquement 0.06% des points originaux, on se demande si l'interpolation RBF ne pourrait pas être utilisé dans le cadre de la compression de données. Il serait intéressant de mener des expérimentations avec des signaux audio ou des photographies en échantillonnant des exemples et en reconstruisant le signal à l'aide de fonctions radiales. En effet, il ne serait alors nécessaire de stocker uniquement les points d'interpolation et les poids $(\alpha_i)_{i \leq n}$.

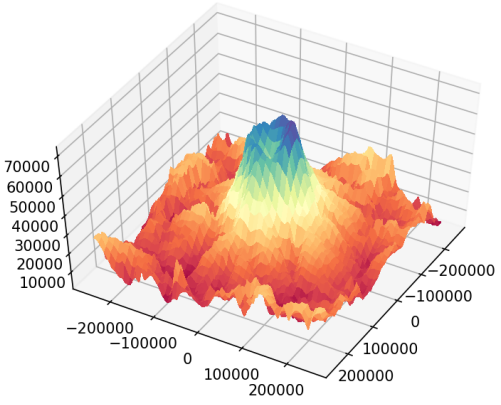
3.3.2 Reconstruction d'un paysage volcanique

On s'intéresse maintenant à la reconstruction d'un paysage montagneux volcanique composé de 160 801 points et qui est particulièrement intéressant car beaucoup moins lisse que le visage de la partie précédente. On trace l'échantillon comme précédemment dans la Figure 26 et on essaye ensuite d'interpoler une solution à l'aide des méthodes utilisées jusqu'ici. Il apparait déjà assez évident que nous allons devoir utiliser un paramètre de forme $\epsilon = 8$ beaucoup plus élevé au vu de l'irrégularité du terrain qui contraste avec le caractère lisse du visage humain. On se servira d'une fonction multiquadratique pour réaliser l'interpolation présentée dans la Figure 27.



(a) Echantillon de 16 080 points (b) Surfaces triangulaires sur 16 080 points

Figure 26: Echantillon de points décrivant un paysage volcanique



(a) $\epsilon = 8$

Figure 27: Interpolation sur un échantillon de 3216 points décrivant un paysage volcanique

4 Application : résolution d'équations aux dérivées partielles

La première application que nous voulons proposer concerne la résolution d'équations aux dérivées partielles à l'aide d'interpolation à base de fonction radiales. De nombreuses méthodes d'analyse numériques existent pour la résolution d'équations aux dérivées partielles (différences finies, éléments finis et volumes finis) et requièrent souvent l'établissement d'un maillage plus ou moins régulier afin de résoudre l'équation. La méthode que nous allons introduire est dite "meshless" au sens ou elle peut se faire avec des points choisis arbitrairement dans le domaine d'étude. Cela présente de nombreux avantages, que ce soit pour étudier des problèmes posés sur des domaines aux formes atypiques ou dynamiques (le domaine évolue au fur et à mesure du temps).
Nous utiliserons principalement la méthode de Kansa dans les parties suivantes, en supposant que la solution de notre système différentiel peut être décrite comme une somme finie de fonctions radiales.

4.1 Equation de poisson

Dans cette partie, nous cherchons à résoudre l'équation suivante :

$$\begin{cases} \Delta u(x) = f(x), & \text{sur } \Omega \\ u(x) = g(x), & \text{sur } \delta\Omega \end{cases} \tag{1}$$

où $\Omega = [0,1]^2$, $u : \mathbb{R}^2 \to \mathbb{R}$ et $\triangle$ est l'opérateur laplacien ($\triangle \cdot = \text{div}(\nabla \cdot)$). On suppose que l'on connait la valeur de f et de g uniquement en un certain nombre de points du domaine. On prend $f(x,y) = \frac{1}{2}(x^2 + y^2 - x - y)$ et $g(x,y) = 0$ afin de disposer d'une solution analytique au problème différentiel $u(x,y) = \frac{1}{4}(xy(x-1)(y-1))$, représentée dans la Figure 28, pour pouvoir estimer l'erreur de l'interpolation.

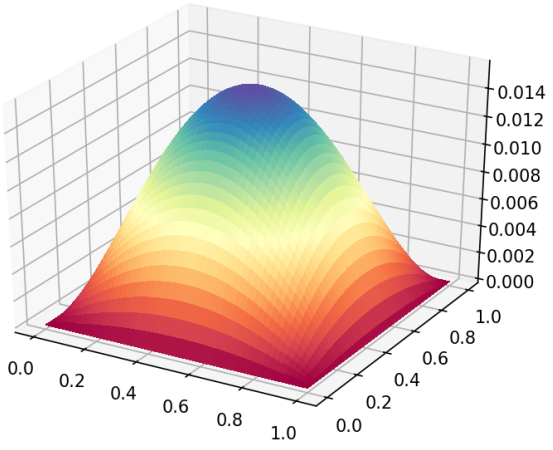


Figure 28: Solution exacte u de l'équation de Poisson

Nous prendrons des points tirés aléatoirement sur la frontière et à l'intérieur du domaine, ce qui nous permet d'obtenir deux ensembles distincts, $\mathcal{X}_\Omega = (x_1, \ldots, x_{N_1})$ et $\mathcal{X}_{\delta\Omega} = (x_{N_1+1}, \ldots, x_{N_2})$, respectivement de taille $N_1 = 10$ et $N_2 = 100$, représentés dans la Figure 29.

Ensuite, on part du principe que l'on peut exprimer la solution u comme une somme de $N = N_1 + N_2$ fonction radiales centrées dans les points de la discrétisation précédente :

$$\Phi(x) = \sum_{i=1}^{N} \alpha_i \phi_{x_i}(x) = \sum_{i=1}^{N} \alpha_i \varphi(\|x - x_i\|)$$

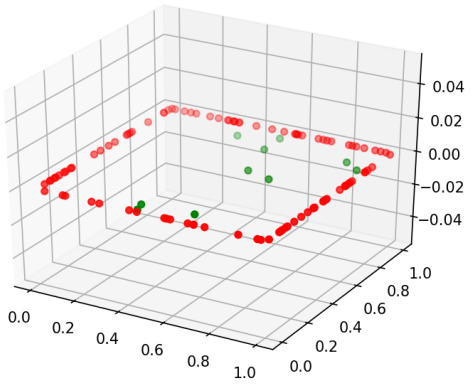


Figure 29: Discrétisation aléatoire de $\mathcal{X}_\Omega$ (en vert) et $\mathcal{X}_{\delta\Omega}$ (en rouge)

On obtient ainsi le système suivant :

$$\begin{cases} \Delta\Phi(x_j) = \sum_{i=1}^{N} \alpha_i \Delta\phi_{x_i}(x_j) = f(x_j), \forall x_j \in \mathcal{X}_\Omega \\ \Phi(x_j) = \sum_{i=1}^{N} \alpha_i \phi_{x_i}(x_j) = g(x_j), \forall x_j \in X_{\delta\Omega} \end{cases}$$

Pour effectuer un calcul de $\triangle\varphi$, on choisit d'utiliser la fonction multiquadratique $\varphi(r) = \sqrt{r+\epsilon^2}$ ou on rappelle que $r_c(x,y) = \|(x,y)-(c_1,c_2)\| = \sqrt{(x-c_1)^2+(y-c_2)^2}$ dans le cas où $\Omega = [0,1]^2$ avec $c = (c_1,c_2)$ est le centre de la fonction radiale. Ainsi, on peut calculer $\triangle\phi_c(x,y) = \triangle\varphi(r_c(x,y))$:

$$\begin{aligned} \frac{\partial\phi_c}{\partial x}(x,y) &= \frac{x-c_1}{\sqrt{r_c(x,y)^2+\epsilon^2}}, & \frac{\partial\phi_c}{\partial x}(x,y) &= \frac{y-c_2}{\sqrt{r_c(x,y)^2+\epsilon^2}} \\ \frac{\partial^2\phi_c}{\partial x^2}(x,y) &= \frac{(x-c_1)^2+\epsilon^2}{(r_c(x,y)^2+\epsilon^2)^{3/2}}, & \frac{\partial^2\phi_c}{\partial x^2}(x,y) &= \frac{(y-c_2)^2+\epsilon^2}{(r_c(x,y)^2+\epsilon^2)^{3/2}} \end{aligned} \tag{2}$$

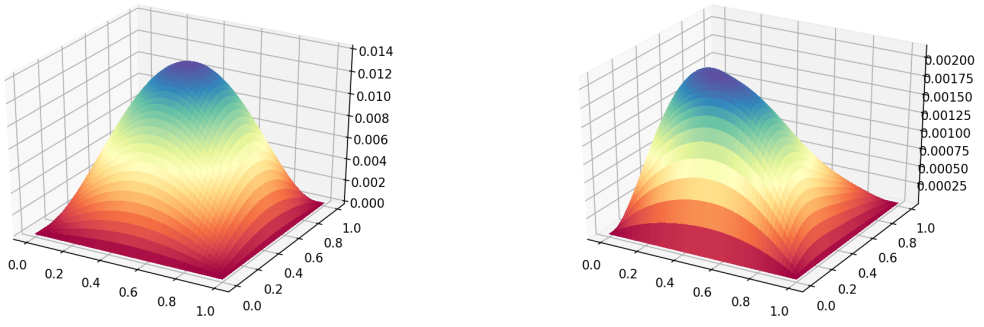
Ainsi, on obtient :

$$\begin{aligned} \Delta\phi_c(x,y) &= \Delta\varphi(r_c(x,y)) \\ &= \frac{\partial^2\phi_c}{\partial x^2}(x,y) + \frac{\partial^2\phi_c}{\partial x^2}(x,y) \\ &= \frac{r_c(x,y)^2+2\epsilon^2}{(r_c(x,y)^2+\epsilon^2)^{3/2}} \end{aligned} \tag{3}$$

Et on peut réécrire le système différentiel sous forme matricielle :

$$\begin{pmatrix} \Delta\phi_{x_1}(x_1) & \Delta\phi_{x_2}(x_1) & \cdots & \Delta\phi_{x_{N_1}}(x_1) & \Delta\phi_{x_{N_1+1}}(x_1)\cdots & \Delta\phi_{x_N}(x_1) \\ \Delta\phi_{x_1}(x_2) & \Delta\phi_{x_2}(x_2) & \cdots & \cdots & \cdots & \Delta\phi_{x_N}(x_2) \\ \vdots & & \ddots & & & \vdots \\ \Delta\phi_{x_1}(x_{N_1}) & \Delta\phi_{x_2}(x_{N_1}) & & \cdots & & \Delta\phi_{x_N}(x_{N_1}) \\ \phi_{x_1}(x_{N_1+1}) & \phi_{x_2}(x_{N_1+1}) & & \cdots & & \phi_{x_N}(x_{N_1+1}) \\ \vdots & & \ddots & & & \vdots \\ \phi_{x_1}(x_{N-1}) & & \cdots & & \phi_{x_{N-1}}(x_{N-1}) & \phi_{x_N}(x_{N-1}) \\ \phi_{x_1}(x_N) & & \cdots & & \phi_{x_{N-1}}(x_N) & \phi_{x_N}(x_N) \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{N_1} \\ \alpha_{N_1+1} \\ \vdots \\ \alpha_{N-1} \\ \alpha_N \end{pmatrix} = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{N_1}) \\ g(x_{N_1+1}) \\ \vdots \\ g(x_{N-1}) \\ g(x_n) \end{pmatrix}$$

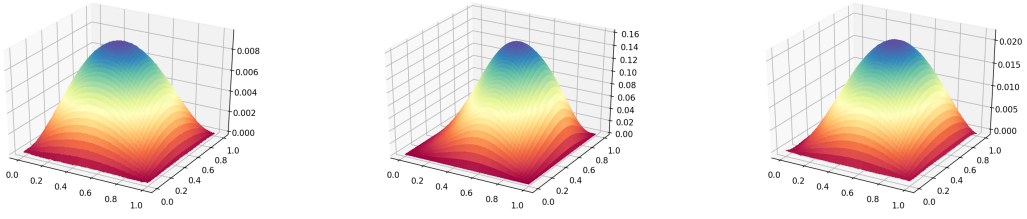
En résolvant le système linéaire, on obtient alors les coefficients de la solution interpolée que l'on représente dans la Figure 30 avec l'erreur en valeur absolue $|\Phi - u|$, estimée en chaque point d'une grille régulière sur tout le domaine.



(a) Solution interpolée (b) Erreur en valeur absolue

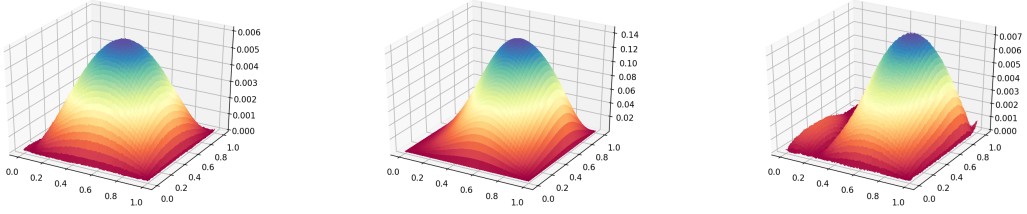
Figure 30: Interpolation de l'équation de poisson sur un domaine rectangulaire

On observe alors qu'on a un résultat assez précis au vu de la solution exacte. Cependant, il faut préciser qu'il a fallu paramétrer l'interpolation pour obtenir ce résultat. En effet, nous avons pris un paramètre de forme ϵ égal à 2.4 et $N_1 = 10$, $N_2 = 100$, mais une légère modification de ces paramètres ne nous permet plus d'avoir d'aussi bon résultats, comme on peut le montrer dans la Figure 31 ainsi que les erreurs représentées dans la Figure 32.



(a) $\epsilon = 4$ (b) $\epsilon = 0.5$ (c) $N_1 = 50,\ N_2 = 100$

Figure 31: Variations des paramètres de formes et de la taille de la discrétisation



(a) Erreur pour $\epsilon = 4$ (b) Erreur pour $\epsilon = 0.5$ (c) Erreur pour $N_1 = 50,\ N_2 = 100$

Figure 32: Erreur en valeur absolue pour une variation des paramètres de l'interpolation

Il apparait comme important de pouvoir poser un cadre théorique à la stabilité du problème en fixant des estimateurs de l'erreur commise en fonction des paramètres du modèle. On pourra

s'inspirer des solution d'estimation d'erreur développées dans [Mar18] afin de l'adapter au cadre de la résolution numériques des équations aux dérivés partielles.

4.2 Equation de la chaleur

On s'intéresse maintenant à l'équation de la chaleur décrite par le système suivant :

$$\begin{cases} \delta_t u(t,x) - \alpha\Delta u(t,x) = f(t,x), & \text{sur } \Omega \times [0,T] \\ u(t,x) = g(t,x), & \text{sur } \delta\Omega \\ u(0,x) = u_0(x) \end{cases} \tag{4}$$

où $\Omega = [0,1]^2$ et $u : [0,T] \times \mathbb{R}^2 \to \mathbb{R}$.
On discrétise l'espace temporel $[0,T]$ avec un pas de temps Δt tel que $u^n(x) = u(n\Delta t, x)$ On introduit le schéma implicite en temps suivant :

$$\frac{u^{n+1}(x) - u^n(x)}{\Delta t} - \alpha\Delta u^{n+1}(x) = f^n(x)$$

Et on exprime la solution u dans la base de fonctions radiales telle que les poids $(\alpha_i)_{0\leq i\leq N}$ seront actualisés à chaque pas de temps, c'est à dire :

$$u^n(x) = \sum_{i=1}^{N} \alpha_i^n \phi_{x_i}(x)$$

On substitue la solution exprimée dans une base de fonctions radiales dans le schéma implicite pour obtenir :

$$\begin{aligned} \frac{1}{\Delta t}\sum_{i=1}^{N}(\alpha_i^{n+1} - \alpha_i^n)\phi_{x_i}(x) - \alpha\sum_{i=1}^{N}\alpha_i^{n+1}\Delta\phi_{x_i}(x) = f^n(x), & \quad \forall x \in \mathcal{X}_\Omega \\ \sum_{i=1}^{N}\alpha_i^{n+1}\left(\frac{\phi_{x_i}(x)}{\Delta t} - \alpha\Delta\phi_{x_i}(x)\right) = \sum_{i=1}^{N}\frac{\alpha_i^n}{\Delta t}\phi_{x_i}(x) + f^n(x), & \quad \forall x \in \mathcal{X}_\Omega \\ \sum_{i=1}^{N}\alpha_i^{n+1}P_i(x) = F^n(x), & \quad \forall x \in \mathcal{X}_\Omega \end{aligned} \tag{5}$$

avec $P_i(x) = \frac{\phi_{x_i}(x)}{\Delta t} - \alpha\Delta\phi_{x_i}(x)$ et $F^n(x) = \sum\limits_{i=1}^{N}\frac{\alpha_i^n}{\Delta t}\phi_{x_i}(x) + f^n(x)$.

On rajoute la condition au bord :

$$\sum_{i=1}^{N}\alpha_i^{n+1}\phi_{x_i}(x) = g^n(x),\ \forall x \in \mathcal{X}_{\delta\Omega}$$

Afin d'obtenir un système matriciel de taille $N \times N$

$$\begin{pmatrix} P_1(x_1) & P_2(x_1) & \cdots & P_{N_1}(x_1) & P_{N_1+1}(x_1)\cdots & P_N(x_1) \\ P_1(x_2) & P_2(x_2) & \cdots & \cdots & \cdots & P_N(x_2) \\ \vdots & & \ddots & & & \vdots \\ P_1(x_{N_1}) & P_2(x_{N_1}) & & \cdots & & P_N(x_{N_1}) \\ \phi_{x_1}(x_{N_1+1}) & \phi_{x_2}(x_{N_1+1}) & & \cdots & & \phi_{x_N}(x_{N_1+1}) \\ \vdots & & \ddots & & & \vdots \\ \phi_{x_1}(x_{N-1}) & & \cdots & & \phi_{x_{N-1}}(x_{N-1}) & \phi_{x_N}(x_{N-1}) \\ \phi_{x_1}(x_N) & & \cdots & & \phi_{x_{N-1}}(x_N) & \phi_{x_N}(x_N) \end{pmatrix} \begin{pmatrix} \alpha_1^{n+1} \\ \alpha_2^{n+1} \\ \vdots \\ \alpha_{N_1}^{n+1} \\ \alpha_{N_1+1}^{n+1} \\ \vdots \\ \alpha_{N-1}^{n+1} \\ \alpha_N^{n+1} \end{pmatrix} = \begin{pmatrix} F^n(x_1) \\ F^n(x_2) \\ \vdots \\ F^n(x_{N_1}) \\ g^n(x_{N_1+1}) \\ \vdots \\ g^n(x_{N-1}) \\ g^n(x_n) \end{pmatrix}$$

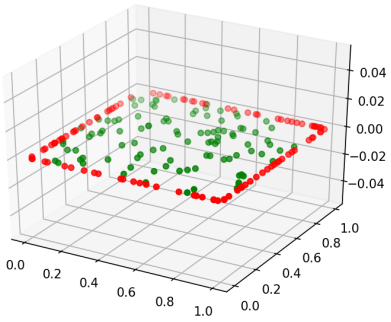
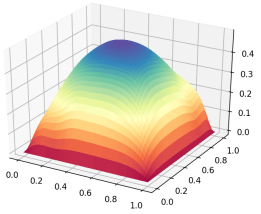
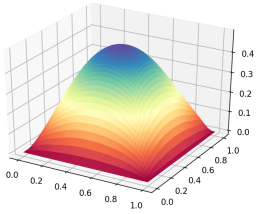


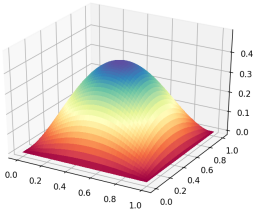
Figure 33: Discrétisation de l'espace pour l'équation parabolique



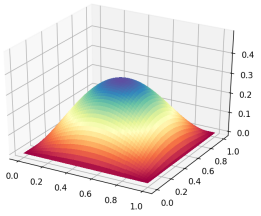
(a) Solution à $t = 0$



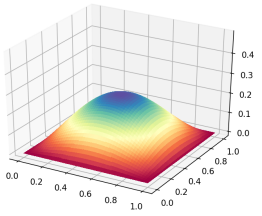
(b) Solution à $t = 0.4$



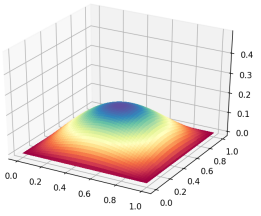
(c) Solution à $t = 0.8$



(d) Solution à $t = 1.2$



(e) Solution à $t = 1.6$



(f) Solution à $t = 2$

Figure 34: Solution de l'équation parabolique

que l'on résout à chaque pas de temps. On prend alors $T = 2$, $\alpha = 1$, $u_0(x) = 1$ et $g(t,x) = f(t,x) = 0$ afin de résoudre l'équation avec un pas de temps $\Delta t = 0.5$. On discrétise l'espace Ω avec des $N_1 = 100$ points intérieurs et $N_2 = 100$ points sur la frontière comme sur la Figure 33 et on résout ensuite le problème d'interpolation dans la Figure 34.

Comme dans le cas de l'équation de Poisson, ce résultat est encore très conditionné à la paramétrisation de l'interpolation, car il suffit de modifier le paramètre de forme $\epsilon = 1$ pour obtenir un résultat totalement incohérent dans le temps. Comme nous l'avons recommandé dans la partie précédente, il apparaît judicieux de construire un cadre théorique d'estimation de l'erreur afin de mieux comprendre le rôle joué par chacun des paramètres du modèle d'interpolation.

5 Application : réseaux de neurones artificiels à base de fonctions radiales

Initialement, les réseaux de neurones se sont inspirés du cerveau pour essayer de reproduire la complexité des raisonnements humains. Ils se sont particulièrement concentrés sur la recherche des configurations optimales pour effectuer des tâches spécifiques comme la vision par ordinateur, le traitement de langage naturel ou la reconnaissance vocale. Nous sommes encore loin d'avoir atteint le potentiel du cerveau humain dans son ensemble mais, dans certaines tâches spécifiques, les réseaux de neurones ont de meilleures performances que les êtres humains.

5.1 Notions fondamentales

Nous allons nous intéresser plus particulièrement à un problème classique de machine-learning : la reconnaissance des chiffres manuscrits. La Figure 35 montre quelques exemples de chiffres manuscrits et la valeur correspondante.

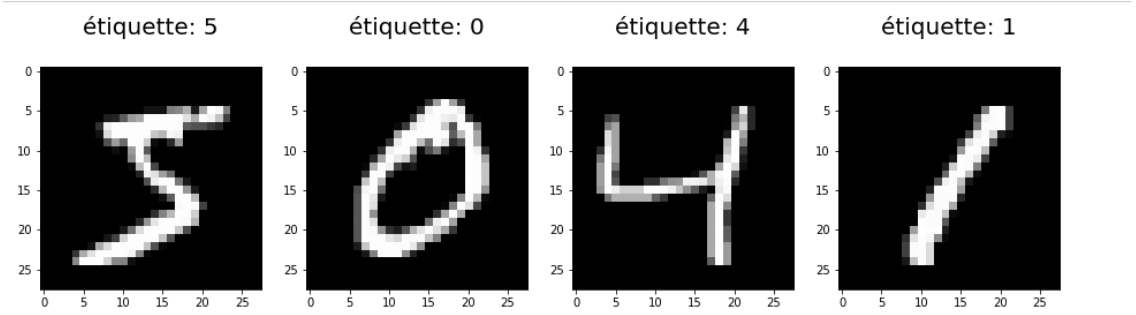


Figure 35: Exemples de chiffres manuscrits

Le réseau de neurones va nous permettre de mettre au point un mécanisme de prédiction du nombre écrit à partir d'une image uniquement, sans la valeur associée. Les images sont données sous la forme de tableaux de dimension 28x28 que nous considérerons comme des vecteurs de taille $p = 784$ avec, pour chaque case, la valeur du pixel en nuances de gris de 0 à 255. On veut donc prédire la classe d'une image quelconque $X \in \mathbb{R}^p$ dans l'ensemble $\{0, 1, ..., 9\}$. C'est ce que nous appelons un problème de classification.

Notons par $\mathcal{X} \subset \mathbb{R}^p$ l'ensemble des images dont nous disposons pour entraîner le réseau de neurones et par $\mathcal{Y}$ l'ensemble des valeurs associées à chaque image. Soit D_n notre ensemble d'apprentissage tel que $(X_1, Y_1), ..., (X_n, Y_n) \in \mathcal{X} \times \mathcal{Y}$, et on suppose que les observations(X_i, Y_i) sont n réalisations indépendantes d'une variable aléatoire (X, Y) de loi inconnue $\mathbb{P}$. L'objectif est de trouver un classificateur f qui prédit $y \in \mathcal{Y}$ à partir de $x \in \mathcal{X}$.

Définition 5.1. On appelle classificateur toute application mesurable $f : \mathcal{X} \to \mathcal{Y}$. L'ensemble des classificateurs est noté $\mathcal{F}$.

Afin de différencier les bons classificateur des moins bons pour un problème donné, il est nécessaire de définir une mesure de qualité du classificateur f. Dans la suite, on utilisera une fonction de coût selon la définition suivante :

Définition 5.2. On appelle fonction de coût toute application $c : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ mesurable telle que

$$\forall y \in \mathcal{Y},\ c(y, y) = 0 \text{ et } \forall y' \neq y,\ c(y, y') > 0$$

La tableau 6 donne quelques exemples de fonctions de coût dans le cas d'un problème de classification, où f désigne une classificateur :

Nous allons nous servir d'une des fonctions de coût les plus utilisées appelée l'entropie croisée et définie ainsi :

$$c(x) = -\sum_{i=0}^{9} p_i(x) ln(\hat{p}_i(x))$$

Nom	c(f(x),y)
0-1	$\mathbf{1}_{yf(x)<0}$
Hinge	$\max(0, 1 - yf(x))$
Hinge2	$\max(0, 1 - yf(x))^2$
Logistique	$\log(1 + \exp(-yf(x)))$

Table 6: Exemples de fonctions de coût pour un problème de classification

avec $p_i(x) = 1$ si l'instance x appartient à la classe i et égale 0 sinon et $\hat{p}_i(x)$ représente la probabilité (obtenue grâce à une fonction *soft max*) que x appartienne à la classe i. Si l'élément x appartient à la classe i, alors une valeur de $\hat{p}_i(x))$ plus proche de 1 indiquera que la classification est correcte. En effet, pour un élément de classe i, la valeur de p_i est égale à 1 et toutes les autres valeurs sont nulles. Donc la fonction d'entropie croisée devient uniquement $-\log(\hat{p}_i(x)))$. La minimisation de cette quantité revient dans ce cas à la maximisation de $\hat{p}_i(x))$ ce qui revient à maximiser la probabilité que x appartienne à la classe i.

L'objectif est alors de fournir un classificateur $f \in \mathcal{F}$ tel que $c(f(X_{n+1}), Y_{n+1})$ est aussi petit que possible *en moyenne*. On introduit donc la notion de risque d'un classificateur :

Définition 5.3. On appelle risque d'un classificateur $f \in \mathcal{F}$ (aussi appelé erreur de prévision) par:

$$\mathcal{R}(f) = \mathbb{E}_{\mathbb{P}}[c(f(x), y)]$$

Définition 5.4. Un classificateur de Bayes f^* est un classificateur optimal s'il minimise le risque pour tous les classificateurs dans $\mathcal{F}$. C'est à dire que, si les données sont tirées sous $\mathbb{P}$, alors :

$$f^* = \arg\min_{f \in \mathcal{F}} \mathbb{E}(c(f(X), Y))$$

Cependant, $\mathbb{P}$ est généralement inconnue. Notre objectif sera donc de minimiser la fonction de risque empirique en fonction de notre échantillon D_n, définie ainsi :

$$\mathcal{R}_{D_n}(f) = \frac{1}{n} \sum_{i=1}^{n} c(f(x_i), y_i).$$

5.2 Architecture d'un réseau de neurones à base de fonction radiales (NN-RBF)

Les réseaux de neurones RBF sont des réseaux que l'on qualifie de (**feedforward**) car ils consistent en une lecture des données d'apprentissage suivie d'une correction des paramètres du modèle que l'on appelle **back-propagation**. Ils s'avèrent être à la fois rapides et efficaces, en particulier pour les problèmes de classification, ce que nous montrerons dans nos résultats. Ils sont composés de 3 couches : une couche d'entrée qui lis les éléments de $\mathcal{X}$, une couche de neurones à base de fonctions radiales, et une couche de sortie qui indique la probabilité qu'un élément de $\mathcal{X}$ appartienne à chaque classe dans $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, comme illustré dans la Figure 36.

Dans la seconde couche chaque neurone consiste en une fonction radiale de base centrée en un vecteur prototype $\mu_j \in \mathcal{X}$. Autrement dit, chaque neurone RBF contient un vecteur prototype qui un élément de l'ensemble d'apprentissage. Il calcule la distance entre son prototype et le vecteur d'entrée, applique la fonction radiale, et envoie le résultat entre 0 et 1 à la troisième couche. Si l'entrée est égale au prototype, alors la sortie du neurone vaut 1. Plus la distance entre le prototype et l'entrée augmente, plus la valeur de sortie décroît exponentiellement vers 0.

La troisième couche est composée d'une fonction appelée *soft-max* qui permet de traiter l'ensemble des valeurs issues des neurones de la seconde couche, de les pondérer et de calculer un vecteur $y \in \mathbb{R}^{10}$ de probabilités ou chaque élément y_i représente la probabilité que le vecteur d'entrée x appartienne à la classe i.

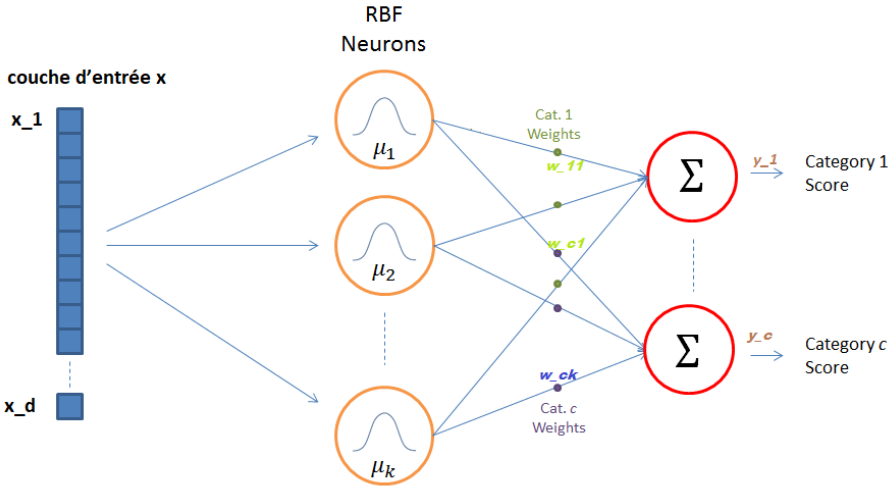


Figure 36: Architecture d'un réseau de neurones à base de fonctions radiales

5.3 Apprentissage par descente de gradient

On appelle *apprentissage* d'un réseau de neurones la procédure qui consiste à estimer les paramètres du modèle à l'aide de l'ensemble des données d'entraînement afin de minimiser la fonction de coût et d'améliorer la précision du réseau. L'apprentissage consiste à ajuster les poids (ainsi que les prototypes) appliquées aux sorties des neurones RBF à l'aide d'une méthode appelée *back-propagation* permettant de modifier les poids en fonction des erreurs de classification commise. Un schéma récapitulatif est proposé dans la Figure 37. Nous allons maintenant détailler les différents algorithmes permettant de minimiser la fonction de coût.

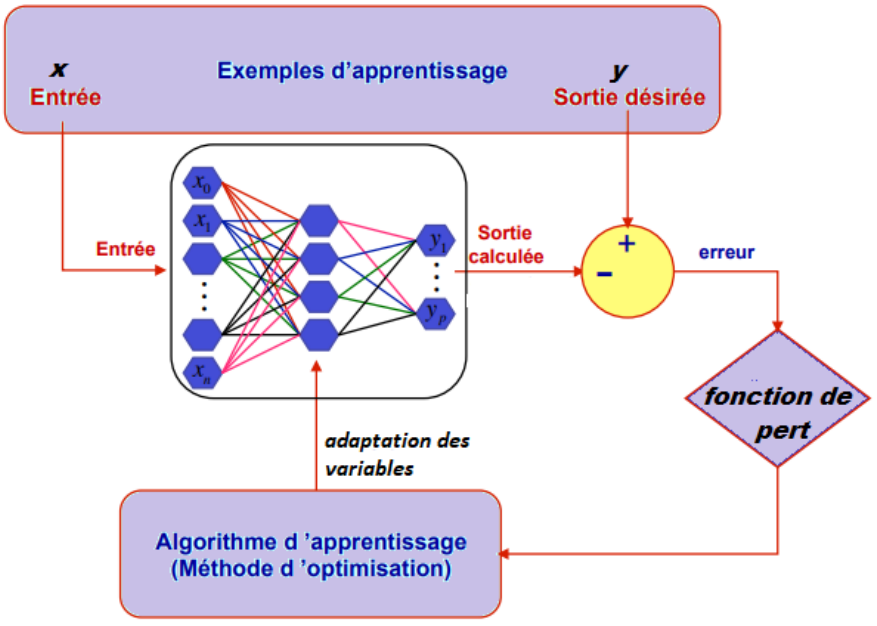


Figure 37: Mécanisme d'apprentissage d'un réseau de neurones

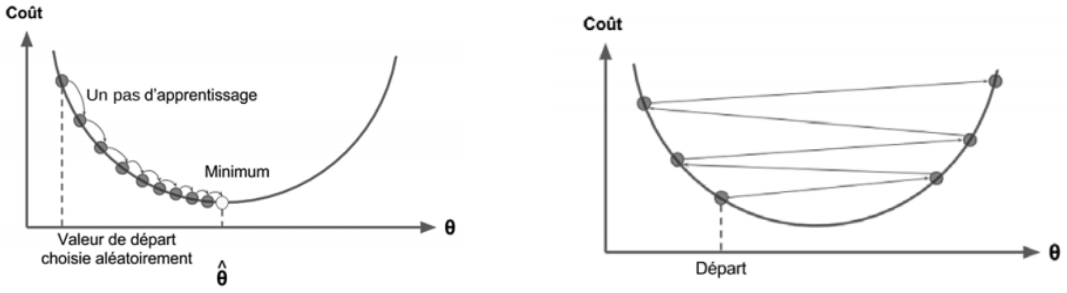
L'apprentissage est l'estimation des paramètres du modèle par minimisation du risque empirique. On note J la fonction de coût que l'on cherche à minimiser (consistant en l'évaluation du coût pour un sous ensemble de $\mathcal{X}$). Un minimum d'une fonction définie sur un espace vectoriel est un point où son gradient s'annule (condition nécessaire d'optimalité). Cependant, on ne peut souhaiter atteindre un tel objectif pour deux raisons : premièrement car la fonction de coût est plutôt complexe et il n'est pas évident de calculer son gradient ou de trouver un ensemble de paramètre qui permet de l'annuler. Deuxièmement, cela pourrait amener à une situation appelée *sur-apprentissage* où le modèle prédirait parfaitement les données de l'ensemble d'entraînement $\mathcal{X}$ mais ne saurait prédire correctement de nouvelles données. Il existe plusieurs approches pour résoudre ce problème dont celle appelée *dropout* détaillé dans [S$^+$14].

Les algorithmes de descente de gradient, introduits initialement par Cauchy en 1847 dans [Cau47], sont des algorithmes itératifs approchant les minimums d'une d'une fonction différentiable définie sur un espace euclidien $\mathcal{E}$ ou un Hilbert $\mathcal{H}$. L'idée fondamentale est de suivre, à chaque étape, la direction de la plus forte pente représentée mathématiquement par l'opposé du gradient. Pour minimiser une fonction J différentiable sur $\mathcal{H}$, l'algorithme s'exprime donc de façon générale sous la forme :

- Initialisation : choisir un point de départ $\theta_0 \in \mathcal{H}$ (éventuellement aléatoirement)
- Itérer : étant obtenu θ_k, déterminer le gradient $\nabla J(\theta_k)$ et renvoyer

$$\theta_{k+1} = \theta_k - \gamma \nabla_\theta J(\theta_k).$$

γ est appelé le taux d'apprentissage, est un hyperparamètre c'est à dire un paramètre du modèle qui doit être fixé par un autre procédé que l'algorithme d'optimisation lui-même. Si γ est trop petit, l'algorithme devra effectuer un grand nombre d'itérations pour converger et prendra beaucoup de temps. Au contraire, si γ est trop grand, l'algorithme risque d'osciller autour du minimum sans jamais l'atteindre, comme illustré dans la Figure 38.



(a) γ correct (b) γ trop grand

Figure 38: Influence du taux d'apprentissage dans la minimisation de la fonction de coût

Il existe trois variantes de l'algorithme de descente de gradient, en fonction de la quantité des données utilisées dans le calcul du gradient à chaque itération : descente de gradient par batch, par mini-batch et descente de gradient stochastique.

1. **Descente de gradient par batch complet :**
   Elle consiste à calculer le gradient pour tous les éléments de $\mathcal{X}$ à chaque itération pour le calcul de $\theta_{k+1} := \theta_k - \gamma \nabla_\theta J(\theta_k)$. Cette méthode est trop lente sur des grandes bases de données.

   Le terme *epoch* fait référence à l'utilisation de l'ensemble de la base de données d'entraînement pour la mise à jour du paramètre. Le nombre d'*epoch* est fixé en fonction du nombre maximum d'itérations de l'algorithme et de la taille des sous-ensemble sur lesquels on calcule le gradient à chaque itération. Dans le cas de la descente par batch complet, le nombre d'*epoch* est donc égal au nombre d'itérations.

2. **La descente de gradient stochastique (SGD) :**

   La mise à jour des paramètres est faite après le calcul du gradient sur chaque $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ individuellement, en tirant aléatoirement le nouvel élément de $X \times \mathcal{Y}$ à chaque itération. Ainsi, on a :

   $$\theta_{k+1} := \theta_k - \gamma \nabla J(\theta_k; x_i, y_i)$$

   Il est donc plus rapide et peut être également utilisé pour un apprentissage en ligne.La convergence possible avant même d'avoir itéré sur l'ensemble des données d'apprentissage.

   En générale, nous souhaitons éviter de fournir les mêmes exemples d'apprentissage à notre modèle car cela pourrait biaiser l'algorithme d'optimisation. Par conséquent, il est souvent judicieux de mélanger les données d'entraînement après chaque *epoch*.Il faut noter que l'algorithme de descente du gradient stochastique effectue des mises à jour fréquentes avec une variance élevée entraînant une fluctuation importante de la fonction objectif ce qui complique la convergence au minimum exact.

3. **La descente de gradient par mini-batch :**
   La descente de gradient par mini-batch combine les deux exemples précédents et effectue une mise à jour pour chaque mini-lot de n éléments de $X \times \mathcal{Y}$, et mélange l'ensemble d'apprentissage aléatoirement à chaque *epoch*. Ainsi, on a :

   $$\theta_{k+1} := \theta_k - \gamma \nabla_\theta J(\theta_k; x_{(i:i+n)}, y_{(i:i+n)})$$

   Dans ce cas on réduit la variance des mises à jour des paramètres ce qui peut conduire à une convergence plus stable et rendre le calcul du gradient très efficace. La taille du mini-batch est également un hyperparamètre du modèle.

Chacune de ces variantes présente des avantages et des inconvénients mais la descente de gradient par mini-batch semble être l'approche la plus efficace dans la plupart des publications sur le sujet. Cependant, cela ne garantit pas une bonne convergence car il y a de nombreux hyperparamètres à régler afin d'optimiser le modèle :

- Choisir un taux d'apprentissage approprié peut être difficile. Un taux d'apprentissage trop faible entraîne une convergence lente, alors qu'un taux d'apprentissage trop élevé peut entraver la convergence et entraîner une fluctuation autour d'un minimum global. Il existent des algorithmes qui tentent d'ajuster le taux d'apprentissage pendant l'apprentissage. Soit en réduisant le taux d'apprentissage selon des paliers prédéfini ou bien lorsque les variations de la fonction de coût dépassent un certains seuil. Plusieurs de ces méthodes sont abordées dans [DCM92].

- Si nos données sont hétérogènes et que certaines classes y_i ont des fréquences plus faible dans l'ensemble d'apprentissage, nous ne voudrons pas appliquer le même taux d'apprentissage à toutes les classes mais effectuer une mise à jour plus importante pour les classes plus rares.

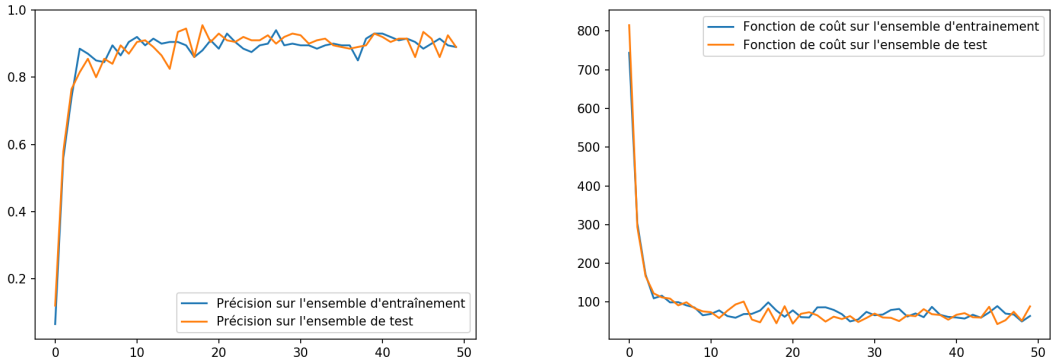
- La minimisation des fonctions d'erreur non strictement convexes consiste à éviter d'être piégé dans leurs nombreux minima locaux. La difficulté provient surtout des points-selle où le gradient est positif dans une direction et négatif dans une autre. Ces points de selle sont généralement entourés par un plateau, ce qui rend difficile pour l'algorithme de descente de gradient d'en sortir car la pente est proche de zéro dans toutes les directions.

Il existe plusieurs algorithmes pour l'optimisation des réseaux de neurones (Momentum, Nestov accelerated gradient, Adagrad etc.). Nous nous servirons d'un algorithme appelé *Adam* (en anglais, *Adaptive Moment Estimation*). Il permet d'adapter le pas d'apprentissage selon les caractéristiques présentes dans les données d'entrée. Pour les caractéristiques fréquentes, le pas d'apprentissage est petit et donc la mise à jour appliquée aux paramètres est amoindrie. Pour des caractéristiques rare (doncplus discriminantes) le pas d'apprentissage est plus élevé.

Nous avons choisi **Tensorflow** , librairie développée par Google, pour l'implémentation de réseau de neurone à base de fonctions radiales. **Tensorflow** est basé sur les tenseurs, une structure de données multidimensionnelle. L'exemple typique de tenseur est un ensemble d'images. Un ensemble d'images est représenté par un tenseur à 4 dimensions: nombre d'images dans l'ensemble, hauteur d'une image, largeur d'une image et nombre de canaux de représentation (3 pour une image en couleurs représentée en RGB, 1 seul pour les images en nuances de gris).

5.4 Résultats

Dans la base de données **MNIST** on dispose de 60 000 échantillons d'entraînement de dimension $p = 28x28 = 784$ et de 10000 échantillons de tests. Cela permet de contrôler les situations de sur-apprentissage. Il existe trois hyperparamètres à définir : la taille du mini-batch n à chaque calcul du gradient, le taux d'apprentissage γ et le nombre de neurones dans la couche intermédiaire m. On commence par montrer l'évolution de la précision et de la fonction de coût dans le cas où $n = 200$, $m = 200$ et $\gamma = 0.05$ avec un nombre d'itérations maximum de 5 000 dans la Figure 39.



(a) Précision sur les deux ensembles

(b) Fonction de coût sur les deux ensembles

Figure 39: Performances du réseau de neurones à base de fonctions radiales

On obtient une une précision de 89% sur l'ensemble de test avec un pic à 95.5%. Afin de se faire une meilleure idée des capacités de notre modèle, on le teste avec différentes configurations d'hyperparamètres et on résume l'ensemble des résultats dans le Tableau 7 en indiquant également le temps de calcul nécessaire à l'entraînement du réseau.

Nombre de centres	Taux d'apprentissage	Taille du batch	Score final	Meilleur score	Temps de calcul
100	0.1	50	0.9	0.96	69.37
		100	0.9	0.94	108.41
		200	0.925	0.935	187.63
	0.01	50	0.84	**0.98**	70.04
		100	0.88	0.96	101.48
		200	0.92	0.935	185.34
200	0.1	50	0.86	0.94	179.09
		100	0.89	0.94	238.37
		200	0.905	0.925	415.4
	0.01	50	0.92	0.96	179.27
		100	0.95	0.96	238.57
		200	0.915	0.94	414.19

Table 7: Résultats du réseau de neurones avec 5000 itérations

On obtient le meilleur score de classification de 98% avec les paramètres suivants $n = 50$, $m = 100$ et $\gamma = 0.01$. On remarque assez directement qu'augmenter le nombre de centre ne permet pas d'obtenir une plus grande précision, malgré la grande dimension (784) des données en entrée. Ensuite, on remarque que le temps de calcul augmente avec le nombre de centres et avec la taille du batch. Afin de réduire ce temps, on essaye d'appliquer une Analyse en Composantes Principales à notre ensemble d'entraînement $\mathcal{X}$ afin d'accélérer le temps de calcul et de pouvoir tester plus d'hyperparamètres. Les résultats sont résumés dans le Tableau 8.

Nombre de centres	Taux d'apprentissage	Taille du batch	Score final	Meilleur score	Temps de calcul
10	0.5	50	0.76	0.84	18.38
		100	0.9	0.94	18.34
		200	0.87	0.92	19.06
		500	0.906	0.932	21.27
	0.1	50	0.88	0.96	18.23
		100	0.86	**0.98**	18.44
		200	0.91	0.945	19.06
		500	0.902	0.942	21.22
	0.01	50	0.86	0.94	18.02
		100	0.91	0.92	18.42
		200	0.96	0.96	19.37
		500	0.908	0.918	21.17
20	0.5	50	0.64	0.68	20.96
		100	0.94	0.94	21.31
		200	0.835	0.92	22.71
		500	0.844	0.884	26.15
	0.1	50	0.92	**0.98**	20.91
		100	0.85	0.97	21.04
		200	0.925	0.96	22.18
		500	0.916	0.94	26.02
	0.01	50	0.88	0.94	20.87
		100	0.89	0.96	20.97
		200	0.93	0.955	22.29
		500	0.922	0.932	26.06
50	0.5	50	0.76	0.88	30.6
		100	0.76	0.87	31.35
		200	0.865	0.905	34.87
		500	0.888	0.926	44.89
	0.1	50	0.88	**0.98**	30.77
		100	0.96	0.97	31.70
		200	0.875	0.95	34.54
		500	0.906	0.946	44.78
	0.01	50	0.88	**0.98**	31.08
		100	0.88	0.96	31.54
		200	0.88	0.96	34.33
		500	0.912	0.938	46.01

Table 8: Résultats du réseau de neurones RBF avec ACP sur 61 composantes et 10 000 itérations

On obtient alors un meilleur score de 98% dans quatres configurations d'hyperparamètres différentes et avec un temps de calcul de seulement 18.44 secondes dans le meilleur des cas. Pourtant, nous avons doublé le nombre d'itérations (de 5 000 à 10 000). Afin de se faire une idée de nos résultats, nous les comparons avec les résultats de Y. LeCun dans sa publication de 1998 [LBBH98] qui a rendu célèbre les réseaux de neurones dans ce *benchmark* classique de la classification des chiffres manuscrits. Avec son équipe de recherche, selon les publications de résultats sur son site internet, ils ont obtenu des scores de classification entre 95.3% et 97.55% de précision sans *preprocessing* des images (l'ACP n'est pas considérée comme tel dans son classement) et en utilisant des réseaux

comportant 2 ou 3 couches intermédiaires avec un nombre de neurones compris entre 300 et 1000. On en déduit que notre modèle, relativement brut comparé aux modèles les plus développés dans ce *benchmark*, est plutôt performant au vu des résultats publiés par les différentes équipes de recherche. Il y a encore de nombreux aspects qui pourraient être améliorés, ne serait-ce que dans le calcul des distances dans lequel nous pourrions éviter la redondance de certains calculs.
Il pourrait aussi être intéressant d'implémenter des couches convolutionnelles à l'aide de fonctions radiales afin de comparer les résultats aux méthodes les plus performantes réalisées par Dan Cireşan, Ueli Meier et Juergen Schmidhuber [CMS12].

6 Conclusion

A travers ce Travail Encadré de Recherche, nous avons eu l'occasion de nous familiariser avec l'interpolation à base de fonctions radiales sous différents aspects. D'abord, nous avons construit théoriquement notre problème d'interpolation afin de nous assurer d'obtenir des résultats dans les applications numériques. Nous avons caractérisé une partie des fonctions radiales garantissant une matrice d'interpolation non-singulière et avons pu mettre en application nos conclusions dans les parties suivantes.

Ensuite, nous avons profité du cadre académique pour tester l'interpolation dans des conditions idéales en générant nos propres échantillons d'interpolation et évaluer l'erreur des interpolants. Cela nous a également permis d'introduire et de discuter plusieurs paramètres du modèle d'interpolation et de présenter quelques idées permettant d'améliorer les résultats pour un échantillon donné. Nous avons aussi pu appliquer les techniques à des échantillons plus complexes dans le cadre de la reconstruction d'un visage ou d'un paysage volcanique.

Nous avons ensuite présenté deux applications correspondant à nos formations au sein du département de Mathématiques : la résolution d'équations aux dérivées partielles et la reconnaissance d'image à l'aide de réseaux de neurones.

Concernant la résolutions des équations aux dérivées partielles, les méthodes à base de fonctions radiales se sont montrées particulièrement précises et efficaces pour résoudre numériquement ces équations. Il faut noter cependant qu'il serait particulièrement intéressant d'appliquer ces méthodes de résolutions sur des équations plus complexes, ainsi que sur des domaines irréguliers : il s'agit en effet d'un des grands avantages des méthodes dites *meshless* telle que celle que nous avons mis en place.
Les réseaux de neurones à base de fonctions radiales se sont montrés très performant et nous ont permit d'obtenir un classificateur reconnaissant les chiffres manuscrits avec une précision proche des publications des meilleurs experts dans le domaine sur des modèles comparables. De plus, nous avons pu réduire le temps d'entraînement à moins de 20 secondes grâce à la combinaison des réseaux RBF et des Analyses en Composantes Principales.

Au final, l'interpolation à base de fonctions radiales est un domaine de recherche qui paraît prometteur au vu de la précision des résultats obtenus et de la vaste étendue des champs d'applications. Il convient néanmoins de remarquer que nous n'avons pas pu aborder en profondeur de nombreux aspects de cette technique d'interpolation qui mériteraient d'être approfondis : l'étude du conditionnement de la matrice d'interpolation, une caractérisation plus exhaustive des fonctions radiales permettant au problème d'interpolation d'être bien posé, différentes méthodes concernant la fixation du paramètre de forme et plus particulièrement celles permettant une paramètre de forme variable, une étude théorique de l'erreur et de la convergence de l'interpolation, la construction d'un cadre théorique plus robuste pour la résolution des équations aux dérivées partielles et l'implémentation de réseaux convolutifs ou récurrents à l'aide de fonctions radiales.

References

[Bar64] Stanley L. Barnes. A technique for maximizing details in numerical weather map analysis. *Journal of Applied Meteorology*, 3(4):396–409, 1964.

[BW03] Mircea Balaj and Szymon Wąsowicz. Haar spaces and polynomial selections. *Mathematica Pannonica*, 14:63–70, 01 2003.

[Cau47] Augustin Cauchy. *Méthode générale pour la résolution des systèmes d'équations simultanées.* C. R. Académie des Sciences de Paris, 1847.

[CMS12] Dan C. Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. *CoRR*, abs/1202.2745, 2012.

[DCM92] C. Darken, J. Chang, and J. Moody. Learning rate schedules for faster stochastic gradient search. In *Neural Networks for Signal Processing II Proceedings of the 1992 IEEE Workshop*, pages 3–12, Aug 1992.

[dT08] Wilna du Toit. Radial basis function interpolation. Master's thesis, Applied Mathematics, Department of Mathematical Sciences, University of Stellenbosch, 2008.

[Fas06] Greg Fasshauer. *Meshfree Methods.* American Scientific Publishers, 2006.

[LBBH98] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[Mar18] Stefano De Marchi. *Lectures on Radial Basis Functions.* Department of Mathematics "Tullio Levi-Civita", University of Padua (Italy), 2018.

[Mer14] Milan Merkle. *Analytic Number Theory, Approximation Theory and Special Functions.* Springer, 2014.

[S+14] Nitish Srivastava et al. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 2014.

[War79] Edward Waring. *Problems concerning interpolation.* Philosophical Transactions of the Royal Society, 1779.