Some contributions of machine learning to quantitative finance : volatility, nowcasting, cva compression

Marc Chataigner

Thèse de doctorat

NNT: 2021UPASM045

université PARIS-SACLAY

Quelques contributions de l'apprentissage statistique à la finance : Volatilité, nowcasting, compression de CVA.

Some contributions of machine learning to finance : Volatility, nowcasting, CVA compression.

Thèse de doctorat de l'Université Paris-Saclay

Ecole Doctorale de Mathématique Hadamard (EDMH) n° 574
Spécialité de doctorat : Mathématiques appliquées
Unité de recherche : Université Paris-Saclay, CNRS, Univ Evry, Laboratoire de Mathématiques et Modélisation d'Evry, 91037, Evry-Courcouronnes, France.
Référent : Université d'Evry-Val d'Essonne

Thèse présentée et soutenue à Évry, le 18 octobre 2021, par

Marc Chataigner

Au vu des rapports de :

Christa Cuchiero Professeure, University of Vienna	Rapportrice
Christoph Reisinger Professeur, Oxford University	Rapporteur

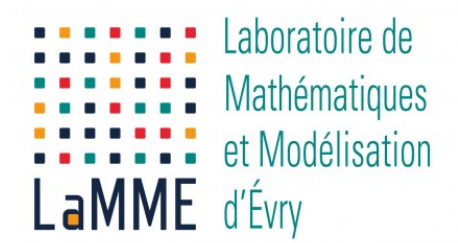
Composition du jury :

Olivier Guéant Professeur, Université Paris 1 Panthéon-Sorbonne	Président
Christa Cuchiero Professeure, University of Vienna	Rapporteure
Christoph Reisinger Professeur, Oxford University	Rapporteur
Agathe Guilloux Professeure, Université d'Évry Val d'Essonne	Examinatrice
Blanka Horvath Maître de conférences, King's College London	Examinatrice
John C. Hull Professeur, University of Toronto	Examinateur

Encadrants :

Stéphane Crépey Professeur, Université Paris-Saclay GS Economie & Management	Directeur
Stéphane Afchain Docteur, HSBC	Invité


université
évry
val-d'essonne
université
PARIS-SACLAY


Laboratoire de
Mathématiques
et Modélisation
LaMME d'Évry


Fondation mathématique
FMJH
Jacques Hadamard

Contents

Remerciements

Cette thèse est l'aboutissement d'un travail de trois années que l'on perçoit trop souvent comme une aventure solitaire mais qui est au contraire un résultat collectif.

Je remercie tout d'abord mon directeur de thèse, Stéphane, pour m'avoir donné ma chance bien avant le début officiel de cette thèse mais surtout pour avoir cru en moi malgré mes nombreux doutes. Cette confiance j'en suis redevable également envers Samuel Drapeau notamment pour avoir appuyé ma candidature à la suite d'un atelier de mathématiques à Shanghai.

Mes remerciements vont également aux rapporteurs Pr Reisinger et Pr Cuchiero pour le temps consacré à la relecture de cette thèse ainsi que les corrections apportées au présent manuscrit. Je suis également reconnaissant envers Mme Horvath, Pr Guilloux, Pr Hull et Pr Guéant d'avoir accepté de faire partie de mon jury de thèse. Leurs remarques enrichissantes durant la soutenance ont ouvert de nouvelles pistes d'amélioration de ces travaux de recherche.

Je remercie aussi HSBC France pour le soutien financier, leur accueil et leur aide matériel à la réalisation de plusieurs travaux de cette thèse.[1]. Les expériences numériques des chapitres 3 et 4 n'auraient jamais pu voir le jour sans les données fournies par HSBC. Je joins à ces remerciements l'institut Europlace de finance pour leur aide financière avant et pendant cette thèse et en particulier Guillaume Macey, Nicolas Grandchamps des Raux et Olivier Guéant pour les échanges constructifs au sein de l'Initiative de Recherche.

Un grand merci également aux équipes quants pour leur accueil chaleureux et leurs précieux conseils en particulier Stéphane (A.), Éric, Hugo, Mahamadou, Eugène, Jérome, Boris, Yann, Côme... Je m'arrête là sinon la liste serait trop longue.

Cette aide matérielle est aussi le fait de l'école doctarale de mathématiques Hadamard qui à travers la bourse du labex LMH m'a permis d'aborder sereinement cette thèse. Par ailleurs le suivi de l'EDMH fut précieux et à ce titre il m'est nécessaire de souligner le travail Pierre Gilles Lemarié-Rieusset, Vincent Sécherre,

[1]The PhD thesis of Marc Chataigner is co-funded by the Research Initiative "Modélisation des marchés actions, obligations et dérivés", financed by HSBC France under the aegis of the Europlace Institute of Finance, and by the public grant ANR-11-LABX-0056-LLH LabEx LMH.

Frédéric Paulin ou encore Clothilde Dépenoux.

Du côté de l'université d'Évry, je souhaite remercier les membres du LaMME pour les séminaires tous aussi intéressants bien que la crise sanitaire et les contraites de temps ne m'aient pas permis de vous rencontrer plus souvent. Merci en particulier à Valérie et Maouloud pour leur aide durant mon installation.

Ces trois années ont donné lieu à des collaborations fructueuses et ce fut un plaisir de travailler avec vous Matthew, Djibril, Aresky et et Jiang. La qualité de ces travaux est aussi le fruit de votre disponibilité et vos conseils lorsque les résultats semblaient insatisfaisants.

Ces remerciements incluent également Bouazza et Hoang pour leurs coopérations lors de projets de recherche ou de travaux dirigés. Je ne doute pas que votre abnégation sera à votre tour récompensée.

Par ailleurs je ne peux pas oublier l'aide apportée par mon parrain, tante Myriame, Solène, Johannes et Rita notamment pour m'avoir logé en région parisienne. Il n'est parfois pas facile de cohabiter avec un doctorant, surtout lors d'un confinement, et je tenais à souligner leur compréhension.

Un grand merci également à mon oncle Jacky pour m'avoir montré la route à suivre (il comprendra) et à ma sœur Noémie pour son appui.

Enfin cette thèse je la dois aussi à ma grand-mère qui me rappelle d'où je viens et à mes parents qui n'ont eu de cesse de me soutenir pour réaliser les études que je souhaitais.

"Le simple est toujours faux. Ce qui ne l'est pas est inutilisable."
Paul Valéry, (Œuvres II, 1942)

Chapter 1

Introduction

1.1 Apprentissage statistique en finance

L'apprentissage statistique (ou Machine Learning en anglais) vise à améliorer un programme informatique avec l'observation de données statistiques (Mitchell 1997). Cette discipline se situe à l'intersection des statistiques et de l'informatique.

Si l'emploi des statistiques en finance n'a rien de nouveau (voir par exemple (Markowitz 1952)), son usage s'est en revanche intensifié ces dernières années pour diverses raisons :

1. Les régulateurs demandent une validation des modèles mathématiques sur données historiques par le biais de backtests (voir par exemple les pages 40-50 de (Basel Committee on Banking Supervision 2011)).

2. La complexité des calculs réglementaires nécessitent l'emploi d'approximations statistiques pour rendre ces calculs accessibles en intra-journalier (cf. section 4.3 de (Crépey, Hoskinson, and Saadeddine 2021)).

3. Les agents financiers souhaitent inclure des données financières non conventionnelles pour dégager davantage de rendement (voir chapitre 5 de (Xing, Cambria, and Welsch 2019) ou encore (Bartram, Branke, and Motahari 2020)).

4. Certains modèles mathématiques ont failli (ou ont été mal utilisé) lors de précédentes périodes de stress financiers. Certains acteurs financiers souhaitent alors relâcher certaines hypothèses sur la dynamique des actifs financiers pour calibrer des modèles plus agnostiques : méthodes non paramétriques (cf. (Chen, Pelger, and Zhu 2020)), volatilité rugueuse (cf. (Gatheral, Jaisson, and Rosenbaum 2018))) ...

L'objectif de cete thèse est de traiter quelques problèmes de la finance quantitative avec des outils issus de l'apprentissage statistique. Les problématiques traitées par la suite s'apparentent aux points 1, 2 et 4 précédents et nous reviendrons dans cette section sur ces points. En particulier nous évoquerons des cas d'applications de la littérature dans les sous-sections 1.1.1 et 1.1.2 pour conclure en discutant quelques fondements des réseaux de neurones.

1.1.1 La quête d'approximations

Un des attraits de l'apprentissage statistique réside dans son évaluation rapide à supposer que celle-ci n'inclut pas la procédure d'apprentissage. Ces approximations statistiques rapides ou "Fast pricing" dans la littérature anglaise font références la plupart du temps à des réseaux de neurones. Leur introduction dans la valorisation de produits dérivés date des années 1990 (cf. (Hutchinson, Lo, and Poggio 1994)) mais leur emploi s'est généralisé bien plus tard et ce pour deux raisons. Tout d'abord la précision de ces approches laissait à désirer, ce qui nécessitait d'affiner l'architecture selon les données (cf. (Garcia and Gençay 2000)). Par ailleurs l'enjeu du temps de calcul n'émerge qu'à partir des années 2010 à cause de modèles de valorisations plus complexes : modèle de Bergomi pour (Horvath, Muguruza, and Tomas 2019) ou calculs de XVA pour (Crépey, Hoskinson, and Saadeddine 2021).

La hausse de la complexité des calculs s'explique par exemple par des valorisations qui incluent des simulations imbriquées. C'est notamment le cas des XVAs (cf. section 1.4) ou d'autres mesures de risques conditionnelles qui rigoureusement appelleraient des simulations de Monte Carlo imbriqués (cf. (Abbas-Turki, Crépey, and Diallo 2018)). Néanmoins les temps de calculs ou plus généralement les contraintes techniques (comme la mémoire des CPUs/GPUs) empêchent la généralisation de ces schémas de simulation pour des XVAs plus complexes comme l'illustre le schéma 1.1.1. Les solutions proposées dans la littérature se basent essentiellement sur des schémas de régression. Ainsi l'espérance conditionnelle est assimilée au prédicteur de Bayes tandis que les quantiles conditionnels sont réécrits sous la forme d'espérance (cf. (Fissler, Ziegel, and Gneiting 2016)).

Le schéma de régression s'apparente à celui de Longstaff-schwartz (voir (Longstaff and Schwartz 2001)) à savoir que l'on simule (et enregistre) dans un premier temps les facteurs de risque jusqu'à une date finale d'une discrétisation temporelle. Puis, à partir de conditions terminales ou payoffs, on régresse les payoffs de la date suivante à partir des facteurs de risque de la date courante. On répète l'opération afin de remonter jusqu'à la date initiale pour obtenir le prix désiré. Traditionnellement la régression était linéaire mais la hausse du nombre de facteurs de risque et surtout la non-linéarité des valorisations des produits dérivés par rapport aux sous-jacents a motivé l'emploi de réseaux de neurones.

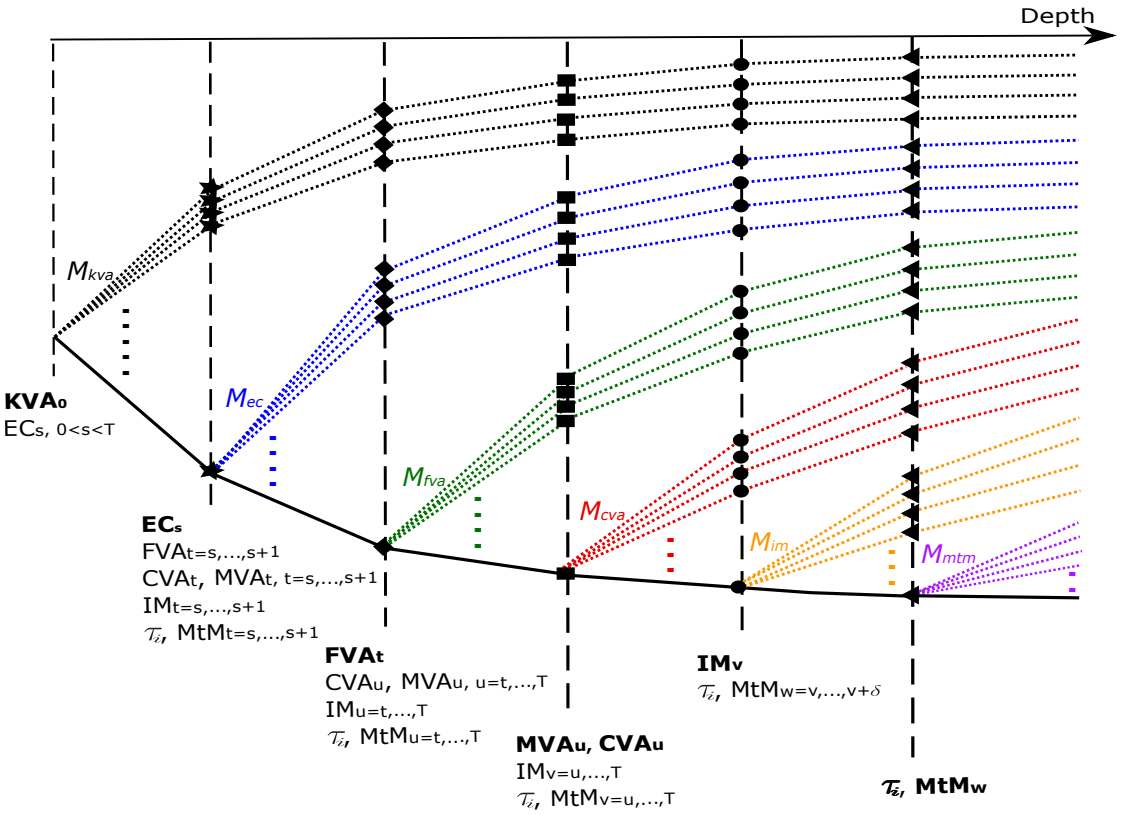


Figure 1.1.1: Arbre de dépendance des XVAs : plus on se déplace à gauche plus la complexité des simulations augmente. La figure est tirée de (Crépey, Hoskinson, and Saadeddine 2021).

Plus généralement les réseaux de neurones sont utiles pour la résolution d'équations aux dérivées partielles (cf. (Lu, Jin, and Karniadakis 2019), (Blechschmidt and Ernst)) car réputés plus résistants face à la malédiction de la dimension.

Ces EDPs peuvent également se reformuler sous la forme d'une équation différentielle stochastique rétrograde (cf. (Weinan, Han, and Jentzen 2017)) dans laquelle le réseau de neurones joue le rôle du gradient de la solution.

Une dernière motivation pour ces approximations rapides est l'emploi de modèles de valorisations plus complexes. On ne cherche pas alors à inclure un apprenant dans un schéma numérique mais à court-circuiter le modèle d'évaluation dans son ensemble (cf. (Ferguson and Green 2018)). Ceci est d'autant plus crucial si l'on souhaite estimer en intra-journalier les sensibilités du modèle de valorisation (ou "pricer" en anglais) à des fins de couverture par exemple. Ceci ce justifie dans des situations où l'implémentation de la différentiation algorithmique peut-être délicate. On peut alors soit estimer une approximation pour obtenir par

différence finie des sensibilités bump-and-revalue ou alors plus simplement estimer ces dérivés partielle par rétro-propagation (cf. (Huge and Savine 2020). La différentiation algorithmique est en effet native dans des librairies d'apprentissage statistique comme Tensorflow (avec des calculs structurés sous forme de graphe) ou pytorch (dont les calculs sont plutôt organisés autour d'un ruban). Ces sensibilités peuvent également servir à calibrer plus rapidement ces modèles d'évaluations complexes (cf. (Horvath, Muguruza, and Tomas 2019)).

1.1.2 Résolution de problèmes encore ouverts

Certains problèmes en finance quantitative ne sont solubles qu'en les formulant sous forme de problèmes d'apprentissage statistique. Une première application consiste à reformuler le problème d'évaluation comme un problème de couverture en marché incomplet c'est-à-dire que le prix n'est plus donné par une espérance mais par la minimisation d'une mesure de risque sur la valorisation du portefeuille couvert. Cette approche a été popularisée entre autres par (Bühler, Gonon, Teichmann, and Wood 2019) qui vise à calibrer, à chaque date de simulation, un réseau de neurones profond pour déterminer à la fois un processus de couverture mais également un prix. La seule restriction porte sur la classe des processus de couverture à travers la paramétrisation du réseau neuronal.

Plus généralement ces réseaux profonds sont utilisés pour résoudre des problèmes de contrôle stochastique sous la forme d'EDSR. À la différence de la sous-section précédente, l'utilisation des réseaux neuronaux ne permet pas seulement l'accélération du temps de calcul mais également leur résolution lorsque le processus de contrôle est déterminé par une classe de fonctions engendrée par la paramétrisation du réseau (cf. (Bachouch, Huré, Langrené, and Pham 2021)).

L'apprentissage statistique permet également d'utiliser des données qui n'étaient pas traitées jusqu'à présent car non standards ou difficiles à formater.

Parmi ces sources de données non conventionnelles, on peut citer des cartes satellitaires pour prédire l'offre de pétrole ou les ventes de commerces de détail ((Katona, Painter, Patatoukas, and Zeng 2018)), l'analyse des réseaux sociaux (sentiment analysis en anglais) pour anticiper une baisse ou hausse des ventes d'une entreprise ((Sprenger, Sandner, Tumasjan, and Welpe 2014)) ou encore le traitement automatisé d'un flux d'actualités pour anticiper le cours d'une entreprise ((Azimi and Agrawal 2019)). Les problèmes financiers sont alors de l'ordre de la prédiction et évalués sur des données historiques.

Enfin l'émergence des réseaux génératifs (GANs, autœncodeurs variationels) ouvre la voie à la génération de scénarios financiers (cf. (Allouche, Girard, and Gobet 2021)). Ces données générées artificiellement peuvent servir pour des tests de résistance (plus communément appelés stress-tests) ou encore pour produire des

jeux de données plus conséquents [1] comme l'expliquent Buhler, Horvath, Lyons, Arribas, Wood, et al. (2020).

1.1.3 L'émergence des réseaux de neurones

L'apprentissage statistique ne se résume certainement pas à l'étude de réseaux de neurones mais ceux-ci répresentent une très large majorité des objets présentés dans ce rapport. Par ailleurs les réseaux de neurones sont l'outil statistique le plus employé par la recherche en finance quantitative et Ruf and Wang (2020) proposent un revue d'articles sur le sujet. Après une brève introduction formelle des réseaux de neurones, nous expliquons l'engouement récent pour cet outil en finance quantitative puis nous donnons l'état de l'art des résultats théoriques de convergence. On se limitera aux réseaux de neurones à propagation avant (ou feedforward en anglais).

Un réseau de neurones à propagation avant est une cascade de régression non-linéaire. S'inspirant du cerveau humain, ce réseau est constitué d'une succession de couches elles-mêmes constituées de neurones. Chaque neurone est composé des éléments suivants :

- Une matrice de poids W qui pondère chacun des neurones de la couche précédente. Des connexions peuvent être ignorées ce qui réduit la taille de cette matrice.
- Un biais b qui en conjonction de la fonction d'activation assure la sparsité du réseau lorsque ce biais est très petit.
- Une fonction d'activation non-linéaire ς qui très souvent joue le rôle de seuil. Dans la cas d'une régression, la fonction d'activation de la couche de sortie (dernière couche) est absente. Si aucune fonction d'activation n'est ajoutée alors on retombe dans une régression linéaire standard.

En résumé un neurone s'écrit comme :

$$f := \varsigma(Wx + b)$$

avec x un vecteur réel retourné par l'entièreté ou un sous-ensemble de la couche précédente si on souhaite ignorer certaines connexions.

En toute généralité ces réseaux prennent en entrée et retournent en sortie un vecteur à valeurs réelles, mais l'utilisateur peut affiner l'image du réseau selon la fonction d'activation de la couche de sortie. Dans le cas d'une régression logistique une fonction sigmoïde permettra d'estimer une probabilité.

[1] La génération ou l'enrichissement de jeux de données sont couramment appelés data augmentation dans la littérature scientifique.

Un réseau de neurones se conçoit dans le cas d'une régression comme une régression linéaire contre un noyau. Ce noyau est constitué des couches inférieures [2] qui agissent comme des fonctions de base non-linéaires et génèrent des variables artificielles (ou features en anglais). La couche de sortie régresse alors linéairement ces variables. Ainsi une régression neuronale se rapproche d'une régression linéaire généralisée mais, à la différence de cette dernière, on apprend conjointement les fonctions de base et la projection sur cette base.

L'apprentissage de ces réseaux repose sur l'algorithme de rétropropagation : le gradient de l'erreur est rétropropagé de la sortie jusqu'aux poids de la couche d'entrée en appliquant la règle de dérivation des fonctions composées. La rétropropagation, qui est un cas particulier de l'AAD [3], est illustrée point par point dans la section 8 de Savine (2018). L'avantage de la rétropropagation est que le calcul de gradient de l'erreur par rapport à (W, b) est du même ordre de grandeur qu'une évaluation de réseau de neurone puisque cette erreur est un scalaire. En revanche si on souhaite calculer le gradient d'une sortie multidimensionnelle du réseau alors le temps de rétropropagation augmente proportionnellement avec la dimension de la couche de sortie.

La rétropagation du gradient permet de calibrer les poids du réseau par descente de gradient. Cette descente est dite stochastique si lors de chaque itération on tire aléatoirement les observations servant à évaluer le gradient. Le souci de l'apprentissage d'un réseau de neurones vient de la non-convexité des problèmes d'apprentissages c'est-à-dire de la fonction de risque par rapport aux poids du réseau. Pour palier à cette non-convexité, les poids W sont initialisés aléatoirement tandis que les biais sont mis à zéro. Néanmoins on doit se contenter d'un minimum local et la difficulté réside dans la sélection de ceux qui conviennent. De plus un grand nombre d'observations est nécessaire pour obtenir une estimation convenable de ce minimum local.

Le succès des réseaux de neurones vient des gains de puissance des machines informatiques avec par exemple l'arrivée des cartes graphiques GPU qui permettent entre autres de manipuler des jeux de données bien plus conséquents. Par ailleurs la médiatisation des travaux en traitement de vidéo et d'image a éveillé la curiosité des acteurs de la place financière. En effet le premier réseau de neurones fonctionnel est réputé être le perceptron de Rosenblatt (1958). L'algorithme de rétropagation n'émerge qu'avec Rumelhart, Hinton, and Williams (1985) et les réseaux convolutionnels avec entre autres LeCun, Boser, Denker, Henderson, Howard, Hubbard, and Jackel (1989). Ce sont ces avancées expérimentales qui ont motivées l'importation des méthodes d'apprentissage profond dans le milieu financier.

[2] c'est-à-dire antérieures à la couche de sortie.

[3] Différentiation algorithmique adjointe

Quelques résultats de convergence permettent l'approximation de n'importe quelle fonction continue par un réseau à une couche caché (ou "shallow network"). Parmi ces résultats on peut citer des théorèmes d'approximation universelle tels que la proposition 2 de Leshno, Lin, Pinkus, and Schocken (1993) qui suppose des fonctions d'activation non polynomiales. D'autre résultats traitent des réseaux profonds parmi lesquels on peut citer (Schmidt-Hieber 2020). Il faut toutefois modérer l'enthousiasme à l'égard de ces résultats puisque le réseau nécessite un très grand nombre de neurones pour atteindre une erreur suffisamment faible. Par ailleurs on n'a pas de contrôle de l'erreur lors de la calibration du réseau sauf cas très rare si on se ramène à un problème d'optimisation convexe (voir théorème 1 et 2 de Vaswani, Bach, and Schmidt (2019)).

1.2 Réseaux de neurones non-arbitrables

1.2.1 Inquiétudes des régulateurs pour ces nouvelles techniques

Dans la section précédente nous avons donné quelques preuves de l'engouement du secteur financier en ce qui concerne l'apprentissage statistique et plus particulièrement l'apprentissage profond. Néanmoins ces nouveaux outils soulèvent quelques inquiétudes du régulateur et pour les référencer nous nous appuyons sur la publication de la banque de france (cf. (Dupont, Fliche, and Yang 2020)).

La banque de France a dégagé quatre critères d'évaluation des algorithmes d'intelligence artificielle :

- La gestion des données avec notamment la confidentialité des informations et le pré-traitement des données ("features engineering").
- La performance de l'algorithme en incluant par exemple la précision du modèle mais également d'autres contraintes potentiellement réglementaires.
- La stabilité de l'algorithme à travers le temps si la distribution des données change à travers le temps ou en généralisant à un nouveau jeu de données et si le modèle est réentrainé.
- L'explicabilité de l'algorithme qui signifie comprendre son comportement, le sens de ses (hyper)paramètres ou encore avoir des preuves de son (dys)fonctionnement.

Dans la suite de ce rapport on ne traitera essentiellement que les trois derniers critères.

Les réseaux de neurones présentent des difficultés pour remplir ces conditions. Leurs poids sont initialisés aléatoirement ce qui affecte la stabilité des réseaux après un entrainement. Par ailleurs leurs paramétrisation riche rend fréquent le surapprentissage il est donc souvent nécessaire de régulariser le critère d'apprentissage comme nous le ferons dans le chapitre 2.

Un critère de convergence en moindre carrés peut se révéler trompeur si les données sous-jacentes sont arbitrables. Il faut donc inclure d'autres critères de performance comme la non-arbitrabilité et pouvoir évaluer a posteriori le respect de ces conditions.

Le choix de l'architecture d'un réseau n'obéit pas à une règle universelle et les paramètres du réseau n'ont pas d'interprétation triviale. Il est alors indispensable de dégager une procédure de sélection des hyperparamètres pour motiver la construction du réseau.

Toutes ces critiques envers l'apprentissage profond provoquent un certain scepticisme des régulateurs comme le montre la figure 1.2.1 de la section 10 de Dupont, Fliche, and Yang (2020). Selon ce schéma, les réseaux de neurones n'offrent pas de

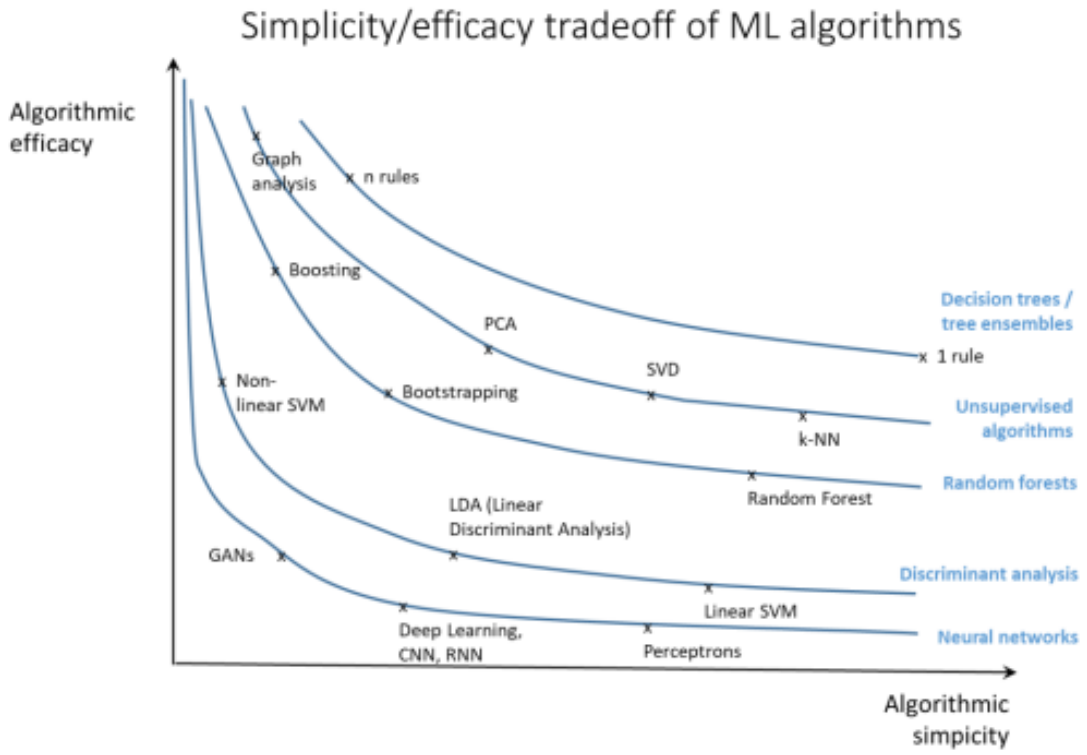


Figure 1.2.1: Compromis simplicité/performance selon la classe d'algorithme d'apprentissage statistique.

compromis entre la complexité du modèle (au sens où l'on saisit son comportement) et sa performance. Un réseau performant est selon le régulateur souvent abscons (phénomène de "boîte noire") et appelle une gouvernance modèle particulièrement rigoureuse.

1.2.2 Contraintes dures versus contraintes souples

La précision d'un algorithme n'est pas une fin en soi comme le souligne la sous-section précédente. Si l'on prend le cas de l'interpolation des prix d'options européennes, des contraintes d'arbitrage doivent impérativement être respectées pour satisfaire aux obligations de contrôle interne des banques. Ces contraintes d'arbitrages se déclinent en contraintes de forme c'est-à-dire portant sur les dérivées partielles de l'interpolateur (notamment vis-à-vis du strike ou de la maturité).

Un réseau de neurones devrait en théorie (voir section 6 de Schmidt-Hieber (2020)) voir ses dérivées partielles converger vers les sensibilités de la fonction de prix. Cependant l'obtention de minimum local de la fonction de risque du réseau n'assure pas en pratique une précision convenable des dérivées du réseau. De plus les données peuvent elles-mêmes violer ces contraintes de forme et il est alors nécessaire de régulariser l'apprentissage du réseau pour ne pas surapprendre.

Il existe alors deux approches pour imposer ces contraintes :

- Les contraintes dures ("hard constraints" en anglais) garantissent le respect de ces conditions quelles que soient les données en entrée du réseau ce qui en interpolation signifie quelle que soit la localisation de l'option.

- Les contraintes souples ("soft constraints" en anglais) pénalisent la fonction de risque si les conditions d'arbitrage sont violées sur des observations de l'ensemble d'apprentissage. Cette approche se traduit en pénalisations dans la routine d'optimisation qui peuvent imposer la non-arbitrabilité sur l'ensemble d'apprentissage mais n'offrent aucune garantie sur de nouvelles observations par exemple dans l'ensemble de test.

Les contraintes dures se traduisent dans le cas d'un réseau de neurones par la construction d'une architecture qui intègre ces contraintes de forme (cf. Dugas, Bengio, Bélisle, Nadeau, and Garcia (2009)). Néanmoins l'architecture induit une classe de fonctions trop restrictive vis-à-vis de la fonction de prix à approcher. Le théorème d'approximation universelle n'est alors plus applicable dans ce cas. Par ailleurs il n'existe pas d'architecture assurant la non-arbitrabilité dans des cas où la condition est plus complexe (voir notamment la condition butterfly[4] pour l'interpolation de volatilité implicite).

[4]Condition qui contrôle les variations spatiales du prix ou plus précisément les dérivées partielles par rapport au strike ou à la log-moneyness.

Les contraintes souples offrent en revanche la souplesse nécessaire pour pénaliser des conditions complexes. L'idée de la pénalisation est d'assurer le respect de conditions de non-arbitrage aux nœuds définis par la grille d'apprentissage. Le pari des contraintes souples est d'exploiter la régularité du réseau pour que ces conditions soient également respectées aux voisinages des nœuds de l'ensemble d'apprentissage.

1.2.3 Autres approximations nonarbitrables

D'autres modèles d'apprentissage statistique peuvent mêler interpolation et respect des contraintes d'arbitrage. Parmi ces modèles on citera les processus gaussiens qui supposent que tous les prix sont distribués selon un vecteur gaussien. La localisation des options, à travers des variables comme la maturité ou le strike, intervient dans le calcul de la corrélation des différents prix. C'est l'objet du noyau qui assigne une corrélation plus forte à des prix dont les coordonnées sont voisines. Un calcul gaussien (voir la sous-section 15.2.1 de Murphy (2012b)) permet ensuite d'interpoler le prix de nouvelles options en exploitant leur corrélation avec les observations d'une grille d'apprentissage. Cousin, Maatouk, and Rullière (2016a) proposent une méthode pour construire des processus gaussiens respectant les contraintes de formes linéaires et ce de manière dure. L'idée est de simuler des trajectoires d'un processus gaussien tronqué c'est-à-dire de rejeter par exemple des réalisation du processus qui ne respecteraient pas les contraintes de monotonie sur une grille discrète. D'autres contraintes linéaires (d'égalité) sont incorporées en modifiant la distribution du processus gaussien (voir page 14 de Cousin, Maatouk, and Rullière (2016a)).

Aubin-Frankowski and Szabo (2020) modélisent des contraintes de forme pour des estimateurs à base d'espace de Hilbert à noyau reproduisant. Le noyau k dans ce cas joue le rôle d'un changement de variable sensé faciliter l'apprentissage d'un estimateur f . Les auteurs renforcent leurs contraintes de forme du type $Df(x_m)+b \geq 0$ appliquées aux nœuds de l'ensemble de l'apprentissage en ajoutant une constante pour arriver à $Df(x_m) + b \geq \eta$. Cette constante η est calibrée en fonction de la régularité du noyau de façon à garantir le respect des contraintes de formes pour un voisinage de tous les points x_m de l'ensemble d'apprentissage, typiquement :

$$\eta = \sup_{m\in\{1,..,M\},u\in\mathcal{B}_{\|.\|}(0,1)} \|Dk(x_m,.) - Dk(x_m+\delta u,.)\|$$

Cependant ce procédé n'est pas compatible avec des contraintes de formes non-linéaires.

1.3 Traiter des données brutes

1.3.1 Défauts des données intra-journalières

Différents fournisseurs de données (Bloomberg, ICAP, ...) informent en continu les opérateurs de marché des valorisations de divers actifs ou produits financiers. Seulement ces données ne peuvent pas être traitées en l'état demandant des post-traitements supplémentaires. Si ces données sont intra-journalières (c'est-à-dire datent de la séance de marché courante) alors il est très probable que l'information soit incomplète. Certains contrats financiers sont par exemple moins liquides et peuvent ne pas être négociés durant la journée ou seulement dans un volume anecdotique.

Par ailleurs des valeurs renseignées pour le fournisseur de données peuvent être incohérentes ou paraître erronées à l'œil d'un expert. Ceci peut s'expliquer par des raisons opérationnelles (erreur humaine, faible volume d'échange), ou à cause d'observations incohérentes temporellement (certaines données n'ont pas été actualisées). On se propose dans le chapitre 3 de corriger ces flux de données.

Les données que nous traiterons ont pour point commun de se structurer sous la forme d'un tenseur. Par tenseur on entend ici une structure de données indexée selon plusieurs dimensions. Par exemple la volatilité implicite d'options européennes s'organise selon un tenseur d'ordre 2 (une matrice) indexé selon la maturité et le strike. L'indexation joue un rôle important puisqu'elle définit la notion de proximité entre les élément du tenseur. Dans ce rapport nous ne traiterons que des tenseurs d'ordre 2 au plus mais il est tout à fait possible d'observer des tenseurs d'ordre supérieur : les volatilité implicites de swaptions sont agencées selon un cube. De plus l'indexation peut varier au cours du temps au sens où les maturités observées un jour peuvent être différentes le lendemain en valeur et en nombre. On ne peut pas ainsi se contenter d'identifier un élément selon ses rangs d'indices.

1.3.2 Détection de valeurs aberrantes

La définition d'une valeur aberrante (ou outlier en anglais) est controversée puisque sa détection nécessite l'avis d'un expert. Si l'on se réfère à Hawkins (1980), une valeur aberrante est une observation qui diffère suffisamment d'autres observations d'un même jeu de données pour soulever un doute sur sa validité. Une valeur aberrante peut se matérialiser par des données corrompues, incomplètes ou encore atypiques en forme.

Il existe plusieurs approches pour formaliser la détection d'observations anormales parmi lesquelles :

- Une valeur aberrante est une observation significativement distante des autres

selon une métrique calculée à partir des variables de l'échantillon. Une analyse de partitionnement (ou clustering en anglais), basée par exemple sur la méthode des K plus proches voisins (Knorr and Ng 1998), peut ensuite être appliquée au jeu de données en basant la règle décision sur la distance ci-dessus évoquée. Une façon de construire cette distance est d'effectuer un changement de coordonnées à travers une analyse en composantes principales (cf. section 3.3 de (Aggarwal 2017)) ou un noyau (cf. section 3.4.3 de (Aggarwal 2017)).

- Une valeur aberrante peut se modéliser selon un modèle statistique duquel découle une notion de vraissemblance. Une observation atypique se traduit alors par un faible niveau de vraisssemblance. Parmi les modèles statistiques on peut citer le filtre de Kalman (cf. (Ting, Theodorou, and Schaal 2007), (Liu, Shah, and Jiang 2004)) ou des modèles de Markov cachés (section 9.3.3 de (Aggarwal 2017)).

- Une valeur aberrante est une observation ne respectant pas la structure latente de l'échantillon. Dans cette perspective, une représentation en basse dimension (avec une ACP ou un autœncodeur selon Aggarwal (2017)) permet de discriminer une observation anormale selon son erreur de reconstruction.

Dans le chapitre 3 notre détection de valeurs aberrantes s'inscrit selon la troisième approche. Néanmoins les ACP/autœncodeurs sont difficilement réconciliables avec des indexations variables à travers le temps puisque ces projections supposent une dimension des observations fixe à travers le temps.

1.3.3 Compléter des données à indexation variable

La complétion matricielle est une sous-catégorie de la branche des statistiques visant à imputer les valeurs manquantes. La complétion suppose que le jeu de données incomplet possède une structure à rang faible ce qui se justifie si les variables de notre dataset sont fortement liées ou si les observations se structurent autour de catégories. Dans cette partie, on notera $D \in \mathbb{R}^{m\times n}$ une matrice représentant le dataset avec m observations (individus) et n variables. L'hypothèse de rang faible signifie que l'on peut factoriser par décomposition en valeurs singulières (tronquée) la matrice D sous la forme $U\Sigma V$ avec $U \in \mathbb{R}^{m\times r}$, $\Sigma \in \mathbb{R}^{r\times r}$ une matrice diagonale et $V \in \mathbb{R}^{r\times n}$. Plus le rang r est faible et plus les connexions entre les variables/observations sont fortes et meilleure sera l'imputation des valeurs manquantes. Si on introduit des valeurs manquantes à D alors une décomposition en valeurs singulières n'est plus possible et il est alors nécessaire d'estimer (U, V, Σ) avec les valeurs restantes du dataset. Il existe alors plusieurs méthodes pour estimer (U, V, Σ) à partir de jeu de données incomplet telles que

l'algorithme *Alternating Least Square* (Hastie, Mazumder, Lee, and Zadeh 2015) ou encore *softImpute* (Mazumder, Hastie, and Tibshirani 2010).

Toute cette branche de l'apprentissage statistique est motivée par les systèmes de recommandations des entreprises du numérique et du marketing en ligne. Les observations du dataset représentent des utilisateurs d'une plateforme et les variables sont des notes de produits vendus en ligne. La complétion permet alors au propriétaire de la plateforme d'inférer les notes non renseignées pour proposer à ses clients des produits qui leurs conviendraient. Ce cas d'application a donné lieu à une compétition (le prix Netflix) dont le jeu de données fait office de dataset de référence dans la littérature (cf. (Nguyen, Kim, and Shim 2019)). Des revues bibliographiques sont disponibles sur le sujet, comme Li, Huang, So, and Zhao (2019) et Nguyen, Kim, and Shim (2019).

Néanmoins la littérature en complétion ne s'applique pas aux données de volatilitiés implicites d'option étudiées dans le chapitre 3 :

- Les colonnes ne sont pas identifiées à des variables puisqu'une option peut expirer un jour et laisser sa place dans la colonne à une option ayant une autre maturité résiduelle.

- La structure matricielle n'encode pas d'informations au sujet de l'indexation des variables. Or cette information est nécessaire pour définir une notion de proximité entre les variables.

- Seule la dernière ligne, correspondant à la journée de cotation courante, contient des valeurs manquantes. Les autres lignes ayant été observées dans le passé, le nombre de valeurs manquantes est réduit et localisé à la différence des jeux de données d'expérience utilisateur (à la "Netflix").

Le chapitre 3 vise à réconcilier l'imputation des valeurs manquantes avec la gestion de données dont l'indexation varie d'un jour sur l'autre. Une structure de faible rang sera toujours apprise à partir de l'historique en apprenant un décodeur mais l'encodeur sera implicite pour pallier au nombre variable d'entrées. Enfin nous supposerons uniquement des liens entre les variables (volatilités implicites) et non entre les journées de cotation. Ceci s'apparente dans la littérature à une approche "item-based"[5].

[5]Dans la littérature des systèmes de recommandation, les lignes des jeux de données sont généralement identifiées à des utilisateurs (users en anglais) et les colonnes à des notations des produits (items en anglais). Les méthodes de complétion sont alors classées comme des approches "item-based" si elles exploitent les intéractions entre les variables plutôt que les intéractions entre les observations. Dans le cas contraire on parle d'approche "user-based".

1.4 Compression des XVAs

1.4.1 Le tournant de 2008

L'acronyme XVA désigne une famille d'ajustements de risques de contrepartie et plus littéralement le X désigne une lettre muette précédant "Value Adjustments". On classera ces primes de risque en trois grandes catégories : les primes de risque de défaut, les primes dénotant le coût de financement du collatéral et les primes de financement de capital réglementaire.

Ces ajustements visent en partie à modéliser deux types de pertes :

- La dépréciation sur le marché d'un contrat financier ne se justifiant que par une perte de confiance dans la solvabilité de la contrepartie. La perte de valeur n'est donc pas reliée aux caractéristiques intrinsèques du contrat.
- Le défaut de la contrepartie qui échoue à honorer ses engagements. Les paiements du contrat échéant après la date de défaut sont alors modélisés comme des flux à couvrir au titre du risque de contrepartie.

La crise financière de 2008 a matérialisé les pertes occassionées par le risque de contrepartie. Tout d'abord le nombre de défauts a augmenté en particulier dans le secteur des crédits hypothécaires américains impactant les marchés financiers par le biais de la titrisation. Un effet de contagion provoque une hausse des spreads de crédit sur l'ensemble des marchés financiers et une pression à la baisse sur les produits dérivés de peur de nombreuses faillites. Par ailleurs le financement des banques est fragilisé : les collatéraux ne sont acceptés qu'en contrepartie de rabais plus important, les taux interbancaires flambent avec un écart de 3,5 % avec l'OIS[6]. La valeur des portefeuilles dérivés des banques étant dépréciée et les besoins de financement plus nombreux pour faire face aux risques, les banques doivent vendre certains de leurs actifs (actions, obligations) pour restaurer leur ratio de capitaux et rétablir la confiance. Or ces ventes forcées (firesales en anglais) accélèrent la baisse des actifs sur les marchés financiers créant ainsi une spirale négative. Cette crise possède ainsi toutes les caractérisques auxquelles souhaitent répondre les XVAs : valoriser les pertes liées aux défauts, anticiper les coûts de financement de collatéral[7], renforcer les capitaux propres des banques pour passer sans encombres les périodes de stress financiers.

[6] L'overnight indexed swap (ou swap de taux au jour le jour) est considéré comme une approximation du taux sans risque. Ce taux correspond à un swap dont la patte flottante paie le taux au jour le jour par exemple le SOFR aux États-Unis. L'écart LIBOR-OIS est un abus de langage renvoyant à l'écart entre les taux de swap calculés sur le LIBOR et sur le taux au jour le jour. Habituellement le LIBOR (taux interbancaire) était proche du taux au jour le jour mais cette observation a été remise en cause en 2008.

[7] Actif financier donné en garantie.

Pour formaliser ces concepts il est nécessaire de faire la différence entre la valeur intrinsèque[8] d'un contrat U et sa valeur de marché V censée appréhender la confiance dans la contrepartie. La différence $U-V$ valorise ce risque de contrepartie et fait référence à la CVA et la FVA que l'on décrira par la suite. Les banques inscrivent dès lors ces quantités à leur bilan comptable pour affiner la valorisation de leurs produits et mettre en place des stratégies de couverture.

Dans cette section, nous présenterons brièvement, pour chaque ajustement de contrepartie, les méthodologies de calcul proposées par le régulateur. Puis nous évoquerons une formulation plus quantitative de ces ajustements de contrepartie afin de mettre en valeur les incomplétudes de marché inhérentes à chaque XVA.

1.4.2 Valoriser le risque de défaut

La CVA, ou plus précisément Credit Value Adjustment en anglais, est une prime de risque pour le défaut de la contrepartie. En effet la banque enregistre une perte si la contrepartie engagée dans un contrat envers la banque fait défaut. Cette perte se mesure à l'aune de l'exposition du client envers la banque si celle-ci est négative (i.e. la contrepartie doit de l'argent à la banque).

On recense plusieurs approches pour évaluer une CVA dite comptable c'est-à-dire reflétant le plus fidèlement les pertes occasionnées par le risque en question. On se limitera au cas unilateral c'est-à-dire la situation ou seul le défaut de la contrepartie est considéré.

La première approche, dite de semi-réplication (Burgard and Kjaer 2011), se prête davantage à la résolution par EDP. Le terme semi fait référence au caractère imparfait de la couverture : le processus de recouvrement R [9] est inconnu et il n'existe pas sur le marché d'instruments de couverture permettant une réplication parfaite du défaut de la contrepartie. D'autres méthodologies ont été proposées pour valoriser des XVAs. On peut notamment citer Albanese, Crépey, and Chataigner (2018) qui modélise le bilan complet de la banque. Ceci permet de calculer la FVA pour un groupe de contreparties et de bénéficier d'un financement mutualisé du collatéral. Par ailleurs la KVA n'est plus considérée comme une dette (liability en anglais) de la banque mais est intégrée à ses capitaux propres ce qui signifie que la KVA n'intervient pas dans le compte de pertes et profits (abrégé P&L en anglais) lié à la contrepartie.

La formule pour la CVA ci-dessous est démontrée dans la section 9.4.1 de Green (2015) dans ce cadre de semi-réplication. Dans le chapitre 4, la métrique optimisée sera calculée selon cette formule (1.1).

[8] La valeur intrinsèque ("clean value" en anglais) ne dépend que des caractéristiques du contrat mais ne fait pas référence à l'identité des signataires.

[9] Pourcentage de l'exposition perdu au moment du défaut.

$$\text{CVA} = (1-R)\int_0^T \lambda_{C\,s} e^{-\int_s^t r_u + \lambda_{C\,u} du} \mathbb{E}\left[(V_s - X_s)^+\right] ds \tag{1.1}$$

avec X_s le montant de collatéral posté à la date s, T la maturité du portefeuille et λ_C le spread de crédit de la contrepartie.

En réalité cette formule est le fruit d'une hypothèse simplicatrice qui suppose l'indépendance entre le montant de l'exposition $V - X$ et le risque de défaut de la contrepartie c'est-à-dire λ_C. Ainsi dans Albanese, Crépey, and Chataigner (2018), le risque de défaut devrait se modéliser plus généralement selon la formule suivante:

$$\text{CVA} = (1-R)\mathbb{E}\left[(V_\tau - X_\tau)^+ \mathbb{1}_{\tau \leq T}\right] \tag{1.2}$$

ou τ désigne le temps de défaut de la contrepartie.

Il est intéressant de s'attarder sur l'exécution opérationnelle de ce genre de formule. La valeur sans risque de contrepartie V (clean value en anglais) peut s'enregistrer suivant un tenseur d'ordre 3 que l'on appelle cube de valeur sans risque de défaut (mark-to-market cube en anglais). Il contient toutes les valeurs des transactions pour toutes les trajectoires d'exposition, toutes les dates jusqu'à maturité. L'enjeu pour l'équipe de la banque en charge du risque de crédit est de valoriser un cube de valeur sans risque de défaut qui soit cohérent avec la méthodologie de valorisation de l'équipe risque de marché. Cette équipe de risque de marché, qui valorise et couvre les transactions indépendamment du signataire, doit s'entendre avec l'équipe risque de crédit sur l'environnement qu'elle utilise (données de marchés, calibration des paramètres modèles) afin d'accorder sa stratégie de couverture du risque de marché avec la couverture des XVAs de l'équipe risque de crédit. En effet l'exposition joue un rôle central dans l'évaluation des XVAs et affecte leurs sensibilités. Dans le cas des CVAs, l'exposition affecte la couverture par rapport à des instruments de marché n'appartenant pas à la classe crédit comme des courbes de taux par exemple. Cet échange rigoureux et automatisé d'information est désigné dans le chapitre 4 sous le nom de ségrégation des desks[10]. Il est d'ailleurs implicitement encouragé par le régulateur qui, dans sa révision des calculs CVA (voir section 4 de Basel Committee on Banking Supervision (2015)) ou avec FRTB-CVA, demande d'inclure la couverture dans le calcul des charges de capital. De plus les CVAs réglementaires devront utiliser des paramètres (drifts, probabilités de défauts, ...) calibrés selon l'approche risque neutre (pages 2-3 de Basel Committee on Banking Supervision (2015)). Le régulateur tend donc à aligner ses calculs réglementaires sur des valorisations comptables (IFRS 13 toujours selon la page 2 du même rapport) qui sont elles alignées avec les valorisations de l'équipe risque de marché.

[10]Un desk est une équipe de trading au sens large.

Enfin la complexité du cube de valeur sans risque de défaut est très élevée et la CVA n'est pas linéaire par rapport au portefeuille. Si la banque conclut une nouvelle transaction avec la contrepartie alors le cube de valeur sans risque de défaut doit être réévalué dans son intégralité pour connaître le nouveau montant de la CVA. Ce contretemps sera limité dans le chapitre 4 grâce au calcul incrémental.

1.4.3 Valoriser le financement du collatéral

On a justifié dans la sous-section précédente qu'une réplication parfaite liée au défaut de la contrepartie est impossible. Dés lors il faut minimiser l'exposition pour réduire le risque résiduel. Une pratique standard du marché est de demander un collatéral en gage de la valeur du portefeuille. Si l'exposition de la banque est négative alors elle doit apporter du collateral et dans le cas contraire c'est à la contrepartie d'apporter le collatéral à la banque. Cet échange de collatéral se produit également entre la banque et les entités auprès desquelles la banque se couvre [11] sauf que la banque poste du collatéral auprès de ces entités si elle reçoit du collatéral de la contrepartie.

Généralement le collateral est un titre financier de bonne qualité comme une obligation souveraine de bonne notation ou simplement des liquidités. Si la garantie est jugée peu fiable alors un rabais est appliqué à la valeur du collatéral et il est nécessaire de mobiliser un nominal plus important en gage.

Il existe plusieurs catégories de collatéral caractérisées par leur mode de calcul:

- La marge de variation (variation margin en anglais) qui est indexée sur le montant du MtM à chaque date de rafraichissement. Le montant posté entre deux appels de marges dépend du montant de collatéral déjà posté et diverses modalités comme des montants minimaux de transfert, des seuils de déclenchement (voir chapitre 6 de (Gregory 2015) pour plus de détails) ... La marge de variation peut être réutilisée comme garantie dans d'autres transactions par le destinataire du collatéral avant l'expiration de la transaction. Ceci signifie que le coût d'emprunt du collatéral pour la couverture est nul si on peut poster le collatéral apporté par la contrepartie. En revanche si la contrepartie ne poste pas de collatéral, parce qu'elle n'a pas signé de credit support annex [12], alors la banque doit emprunter ce collatéral sur les marchés à un coût généralement supérieur à sa rémunération (souvent égale à l'OIS). La Funding Value Adjustment (FVA, pour sa formulation voir chapitre 9 de

[11] Un cas de couverture fréquemment rencontré consiste à ouvrir une position exactement inverse à la position que l'on souhaite couvrir. On parle alors de couverture "back-to-back".

[12] Un credit support annex (CSA en abrégé) est un document juridique qui régie les échanges de collatéral entre les contreparties dans le cadre d'une transaction de produits dérivés.

(Green 2015)) correspond au coût de financement de la marge de variation si la contrepartie n'en poste pas ou pas assez.

- La marge initiale (initial margin en anglais) est un coussin de sécurité minimal visant à couvrir le risque de dérapage du MtM entre deux appels de marge[13]. Pour chaque date de raffraichissement, le montant de collatéral est indexé sur une mesure de risque du portefeuille : Value-at-Risk ou un dérivé de VaR à base de sensibilités dans le cas d'une IM SIMM (Standardized initial margin method). Le coût de financement de cette marge initiale constitue la Margin Value Adjustment (ou MVA, voir chapitre 16 de (Gregory 2015) ou l'annexe de (Crépey, Hoskinson, and Saadeddine 2021) pour sa formulation). La marge initiale ne peut pas être en général réutilisé par son destinataire pour d'autres transactions c'est-à-dire qu'elle est séquestrée.

Les coûts de financement interviennent dans d'autres situations comme avec les chambres de compensations (voir (Armenti and Crépey 2017)). Le régulateur encourage d'ailleurs la compensation des transactions de dérivés voire la rend obligatoire (voir section 9.3 de (Gregory 2015)). Les CCPs (acronyme de central counterparty clearing) n'entreront pas dans le cadre de l'étude du chapitre 4.

La collatéralisation permet de réduire, voire d'annuler la CVA et la FVA. En revanche elle peut donner lieu à d'autres coûts comme la MVA.

1.4.4 Valoriser des provisions en capital

Parallèlement à ces écritures comptables, le régulateur a souhaité imposer des réserves en capital aux banques (premier pilier des réglementations de Bâle) afin d'empêcher la mise en place de pyramides de Ponzi. Le principe de ce montage financier est d'entrer dans de nouveaux contrats afin de financer les pertes d'un portefeuille pré-existant. Or les réserves de capital sont chères à lever auprès des actionnaires puisqu'ils demandent en général un rendement bien supérieur au coût de financement sur le marché interbancaire par exemple. La mise en place d'une transaction est alors plus coûteuse et empêche la contruction d'une pyramide de Ponzi. Les réserves de capital réglementaires sont avant tout conservatives et ne visent pas à couvrir une perte de manière exacte mais plutôt à constituer un coussin de sécurité en prévision de périodes de stress.

Parmi les réserves de capital on peut citer :

- La charge en capital au titre du risque de crédit qui provisionne le défaut des clients de la banque.

[13] Ce risque de dérapage est valorisé sous le terme de Margin period of risk.

- La charge en capital au titre de la CVA qui provisionne des dépréciations de la valeur des actifs de la banque si la signature de ses débiteurs se dégrade.
- La Risk-weighted asset (ou actifs à risques pondérés) qui sert de référence pour provisionner les fonds propres réglementaires de la banque (ratio de Bâle). Tout le bilan de la banque et toutes les classes de risque se veulent capturés par la RWA.

Le détail des formules de chaque réserve de capital peut être trouvé dans (Basel Committee on Banking Supervision 2015) (ou le chapitre 8 de (Gregory 2015)) mais on peut dégager quelques similitudes. Tout d'abord le régulateur propose au moins deux méthodologies de calculs pour s'adapter aux capacités financières de chaque établissement. Parmi ces approches il y a toujours une méthode paramétrique, plus simple à calculer, mais plus conservative. Une autre approche, plus complexe, autorise l'utilisation de modèles d'évaluation internes aux banques ce qui nécessite bien souvent de simuler l'exposition comme pour les autres XVAs.

Par ailleurs toutes ces quantités désignent des montants de capital à réserver mais c'est leur coût de financement qui est chargé au client. La capital value adjustment (KVA) vise à estimer ce coût pour ensuite l'ajouter à la valeur sans risque de défaut V du contrat au côté de la CVA et la FVA. Toutefois la réserve en capital intervenant dans la KVA n'est pas nécessairement basée sur une des réserves de capital réglementaire précédentes mais peut alternativement être basée sur le capital économique. Le capital économique est une mesure de risque désignant quel montant la banque doit posséder en réserve pour traverser une crise. Il s'agit généralement d'une Value-at-Risk[14] ou d'une expected shortfall[15] calculée sur les pertes à l'horizon d'une année de la banque. En notant h le taux de rendement pour les actionnaires des capitaux placés dans la banque, Crépey, Hoskinson, and Saadeddine (2021) dans leur annexe formulent la KVA comme :

$$KVA = \mathbb{E}\left[\int_0^{T\wedge\tau} he^{-hs}\max\left(EC_s, KVA_s\right)ds\right] \tag{1.3}$$

La KVA capture ainsi toutes les pertes issues du risque de contrepartie qui ne peuvent pas être couvertes. Au même titre que la CVA, la KVA et les autres réserves de capital pourraient être optimisées. Toutefois la complexité calculatoire de la KVA rend pour l'instant inaccessible sa compression. À notre connaissance, seule la MVA dans une forme simpliste avec (Kondratyev and Giorgidze 2017) a été l'objet d'une procédure de compression dans la littérature. Nous proposons d'optimiser la CVA et d'en préciser les enjeux opérationnels dans le chapitre 4.

[14]Quantile qui sert à exprimer la pire perte que l'on rencontrerais pour un niveau de confiance fixé (généralement 95%).

[15]Espérance de la Value-at-Risk ayant un niveau de confiance supérieur à un seuil (généralement 97,5%).

Chapter 1

Introduction

1.1 Machine learning in finance

Machine Learning aims at improving a computer program with the observation of statistical data (Mitchell 1997). This discipline is located at the intersection of statistics and computer science.

If the use of statistics in finance is a novel idea (see for example (Markowitz 1952)), its use has intensified in recent years for various reasons:

1. Regulators require validation of mathematical models on historical data through backtests (see for example pages 40-50 of (Basel Committee on Banking Supervision 2011)).

2. The complexity of regulatory calculations requires the use of statistical approximations to make these calculations accessible on an intraday basis (see section 4.3 of (Crépey, Hoskinson, and Saadeddine 2021)).

3. Financial agents want to include unconventional financial data to generate more returns (see chapter 5 of (Xing, Cambria, and Welsch 2019) or (Bartram, Branke, and Motahari 2020)).

4. Some mathematical models have failed (or been misunderstood) during previous periods of financial stress. Some financial players then wish to relax certain assumptions on the dynamics of financial assets in order to calibrate more agnostic models : non-parametric methods (see (Chen, Pelger, and Zhu 2020)), rough volatility (see (Gatheral, Jaisson, and Rosenbaum 2018))).

The objective of this thesis is to treat some problems of quantitative finance with some tools issued from statistical learning. Treated problems thereafter are related to the previous points 1, 2 and 4 and we will examine these points in

greater details in this section. In particular we will evoke cases of applications of the literature in the subsections 1.1.1 and 1.1.2 to conclude by discussing some foundations of neural networks.

1.1.1 The need for surrogate models

One of the appeals of statistical learning is its rapid evaluation, assuming we do not include the learning procedure. These fast statistical approximations or "Fast pricing" in quantitative finance refer most of the time to neural networks. Their implementation for the valuation of derivative products dates back to the 1990s (see (Hutchinson, Lo, and Poggio 1994)) but their use became widespread much later for two reasons. First of all, the precision of these approaches left something to be desired, which called for refining the architecture according to the data (see (Garcia and Gençay 2000)). In addition, the issue of calculation time did not emerge until the 2010s because of more complex valuation models: Bergomi model for (Horvath, Muguruza, and Tomas 2019) or XVA calculations for (Crépey, Hoskinson, and Saadeddine 2021).

The increase in the complexity of the calculations is explained, for example, by valuation procedures that include nested simulations. This is notably the case of XVAs (see section 1.4) or other conditional risk measures which would rigorously call for nested Monte Carlo (cf. (Abbas-Turki, Crépey, and Diallo 2018)) . However, the calculation times or more generally the technical constraints (such as the memory of the CPUs / GPUs) prevent the generalization of these simulation schemes for more complex XVAs as shown in the figure 1.1.1. The solutions proposed in the literature are essentially based on regression schemes. Thus the conditional expectation is assimilated to the Bayes predictor while the conditional quantiles are reformulated in the form of expectation (cf. (Fissler, Ziegel, and Gneiting 2016)).

The regression scheme is similar to that of Longstaff-schwartz (see (Longstaff and Schwartz 2001)) namely that we first simulate (and record) the risk factors until the terminal date of a temporal discretization. Then, the payoffs for the following date are regressed on the basis of the risk factors for the current date. The operation is repeated in order to go back to the initial date in order to obtain the desired price. Traditionally, the regression was linear, but the increase in the number of risk factors and especially the non-linearity of the valuations of derivative products with respect to the underlying has motivated the use of neural networks.

More generally, neural networks are useful for solving partial derivative equations (cf. (Lu, Jin, and Karniadakis 2019), (Blechschmidt and Ernst)) because they are claimed to be more resilient to the curse of dimensionality.

These EDPs can also be reformulated in the form of a backward stochastic

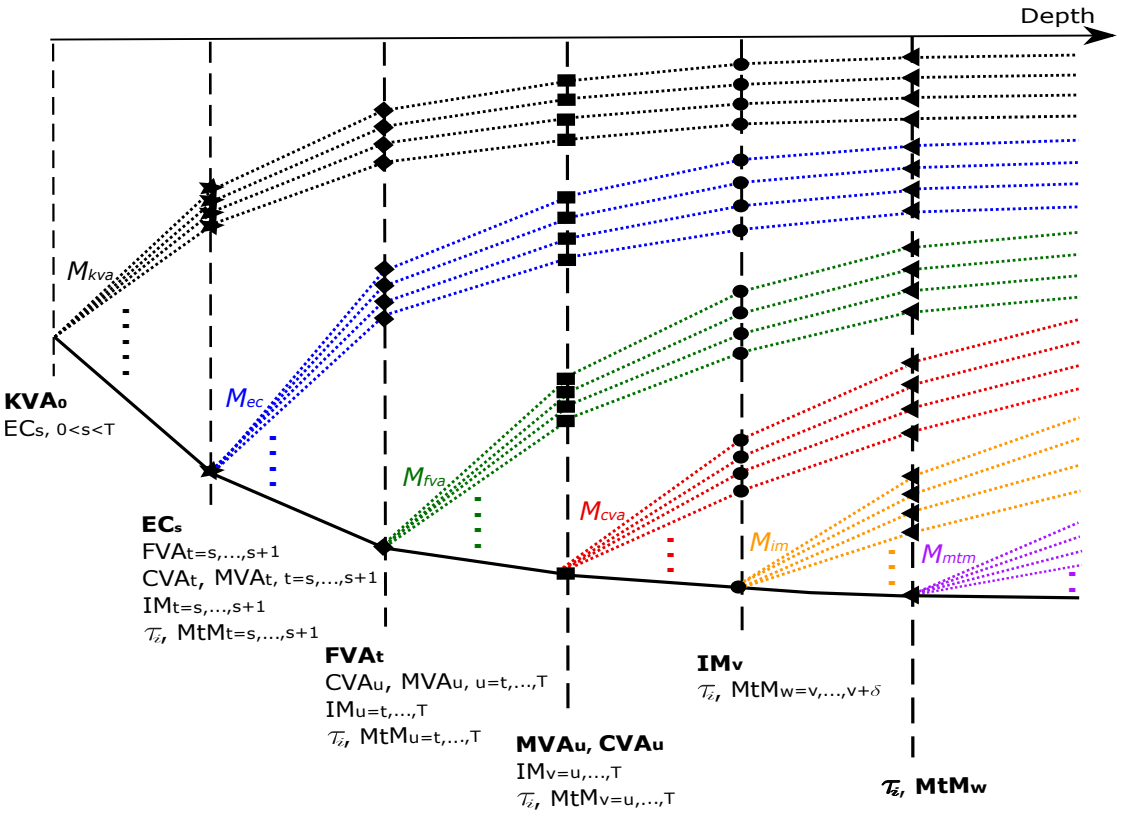


Figure 1.1.1: Dependency tree for XVAs : the more we move to the left, the more the complexity of the simulations increases. Figure is drawn from (Crépey, Hoskinson, and Saadeddine 2021).

differential equation (cf. (Weinan, Han, and Jentzen 2017)) in which the neural network plays the role of the gradient of the solution.

A final motivation for these quick approximations is the use of more complex valuation models. In that situation, we do not try to include a proxy in a numerical scheme but to bypass the pricer as a whole (see (Ferguson and Green 2018)). This is even more crucial if one wishes to estimate on an intraday basis the sensitivities of the pricer for hedging purposes, for example. This is justified in situations where the implementation of algorithmic differentiation can be delicate. We can then either estimate a proxy to obtain bump-and-revalue sensitivities by finite difference or more simply estimate these partial derivatives by back-propagation (cf. (Huge and Savine 2020). Algorithmic differentiation is indeed native in machine learning frameworks like Tensorflow (with calculations structured in as a graph) or pytorch (whose calculations are rather organized around a tape). These sensitivities can also be used to calibrate these complex valuation models more quickly (see (Horvath, Muguruza, and Tomas 2019)).

1.1.2 Solving opened problems

Some problems in quantitative finance can only be solved by formulating them in the form of statistical learning problems. A first application consists in reformulating the valuation problem as an incomplete market hedging problem, that is the price is no longer given by an expectation but by the minimization of a risk measure on the valuation of the hedged portfolio. This approach was popularized among others by (Bühler, Gonon, Teichmann, and Wood 2019) which aims to calibrate, at each simulation date, a deep neural network to determine both a hedging process and also a price. The only restriction concerns the class of hedging processes through the parametrization of the neural network.

More generally, these deep networks are used to solve stochastic control problems in the form of BSDE. Unlike the previous subsection, the use of neural networks not only accelerates the computation time but also allows their resolution when the control process is determined by a function class defined by the network parameterization (see (Bachouch, Huré, Langrené, and Pham 2021)).

Machine learning enables financial companies to use data that was not processed until now because they were non-standard or difficult to format.

These unconventional data sources include satellite maps to predict oil supply or retail sales ((Katona, Painter, Patatoukas, and Zeng 2018)), social media analysis (also called sentiment analysis) to anticipate a decrease/increase in a company's sales ((Sprenger, Sandner, Tumasjan, and Welpe 2014)) or even the automated processing of a newsfeed to anticipate the stock value of a company ((Azimi and Agrawal 2019)). Financial problems are then in the domain of prediction and assessed on historical data.

Finally, the emergence of generative networks (GANs, variational autoencoders) opens the way to the generation of financial scenarios (see (Allouche, Girard, and Gobet 2021)). These artificially generated data can be used for stress tests or to produce larger datasets (data augmentation) as Buhler, Horvath, Lyons, Arribas, Wood, et al. (2020) explains.

1.1.3 The neural network upheaval

Statistical learning cannot be restricted to the study of neural networks, but they represent a very large majority of the objects presented in this report. In addition, neural networks are the statistical tool most used by quantitative finance research and Ruf and Wang (2020) offer a review of articles on the subject. After a brief formal introduction of neural networks, we explain the recent popularity for this tool in quantitative finance and then we give recent theoretical convergence results. We will limit ourselves to feedforward neural networks.

A feedforward neural network is a cascade of nonlinear regressions. Inspired

by the human brain, this network is made up of a succession of layers themselves comprised of neurons. Each neuron contains the following elements :

- A matrix W weighting each of the neurons of the previous layer. Connections can be skipped which reduces the size of this matrix.
- A b bias which in conjunction with the activation function ensures network sparsity when this bias is very small.
- A non-linear activation function ς which very often acts as a threshold. In the case of a regression, the activation function of the output layer (the latest layer) is absent. If no activation function is added then we fall back into an ordinary linear regression.

In summary, a neuron is written as:

$$f := \varsigma(Wx + b)$$

with x a real vector returned by the whole or a subset of the previous layer if we want to skip certain connections.

In general, these networks take as input and return as output a vector with real values, but the user can restrain the image of the network according to the activation function of the output layer. In the case of a logistic regression, a sigmoid function is added to estimate a probability.

A neural network can be seen in the case of a regression as a linear regression against a kernel function. This kernel is made up of the lower layers [1] which act as non-linear basis functions and generate artificial features. The output layer then linearly regresses these features. Thus a neuronal regression approaches a generalized linear regression but, unlike the latter, we jointly learn the basis functions and the projection coefficients on this basis.

The learning of these networks is based on the backpropagation algorithm : the loss gradient is backpropagated from the output to the weights of the input layer by applying the chain rule. The back-propagation, which is a special case of the AAD [2], is illustrated step by step in section 8 of Savine (2018). The advantage of backpropagation is that the gradient computation of the loss relative to (W, b) is of the same order of magnitude as a neural network evaluation since this loss is a scalar. On the other hand, if one wishes to calculate the gradient of a multidimensional output of the network then the backpropagation time increases proportionally with the dimension of the output layer.

The backpropagation of the gradient allows for the calibration of the weights of the network with a gradient descent. This gradient descent is said to be stochastic

[1] That is to say before the output layer.
[2] Adjoint algorithmic differentiation

if, during each iteration, the observations used to evaluate the gradient are sampled randomly. The concern about the learning of a neural network comes from the non-convexity of the learning problems, that is to say from the risk function with respect to the network weights. To overcome this non-convexity, the weights W are initialized randomly while the biases are set to zero. Nevertheless we have to be satisfied with a local minimum and the difficulty lies in the selection of those which are suitable. In addition, a large number of observations is necessary to obtain a suitable estimate of this local minimum.

The success of neural networks comes from the power gains of computer machines with, for example, the arrival of GPU graphics cards which, among other things, allow much larger datasets to be treated. In addition, the progress of video and image processing work has aroused the curiosity of stakeholders in the financial sector. Indeed, the first functional neural network is reputed to be the Rosenblatt (1958) perceptron. The back propagation algorithm only emerges with Rumelhart, Hinton, and Williams (1985) and convolutional networks with, among others, LeCun, Boser, Denker, Henderson, Howard, Hubbard, and Jackel (1989). It is these experimental breakthroughs that have motivated the introduction of deep learning methods into the financial world.

Some convergence results promise the approximation of any continuous function with a shallow neural network. Among these results we can cite universal approximation theorems such as Proposition 2 of Leshno, Lin, Pinkus, and Schocken (1993) assuming non-polynomial activation functions. Other results deal with deep networks among which we can cite (Schmidt-Hieber 2020). However, the enthusiasm for these results must be moderated since the network requires a very large number of neurons to achieve a sufficient accuracy. In addition, we have no error control during network calibration except in very rare cases if we fall on a convex optimization problem (see theorem 1 and 2 of Vaswani, Bach, and Schmidt (2019)).

1.2 Arbitrage-Free neural networks

1.2.1 Criticisms of neural network from a regulatory perspective

In the previous section we gave some evidence of the enthusiasm of the financial sector with regard to statistical learning and more particularly deep learning. However, these new tools raise some concerns from the regulator and in order to reference them we relate our analysis on the publication of the Banque de France (see (Dupont, Fliche, and Yang 2020)).

The bank of France has identified four criteria for evaluating artificial intelli-

gence algorithms :

- Data management including data protection and features engineering.
- The performance of the algorithm including for example the accuracy of the model but also other potentially regulatory constraints.
- The stability of the algorithm over time if data are not stationary (or involve a new dataset) and if the model is retrained.
- The explicability of the algorithm which means understanding its behavior, the meaning of its (hyper) parameters or even having evidence of its (dys)function.

In the remainder of this report, we will mainly deal with the last three criteria.

Neural networks show difficulties in meeting these conditions. Their weights are randomly initialized which affects the stability of the networks after training. Moreover, their rich parameterization makes overfitting frequent, so it is often necessary to regularize the learning criterion as we will do in the chapter 2.

A least square convergence criterion can be misleading if the underlying data is arbitrable. It is therefore necessary to include other performance criteria such as non-arbitrability and we must be able to assess a posteriori compliance with these conditions.

The choice of the neural network architecture is not subject to a universal rule and the parameters of the network have no trivial interpretation. It is thus essential to build a procedure for selecting hyperparameters to motivate the design of the network.

All of these criticisms about deep learning arouse some skepticism from regulators as shown in figure 1.2.1 in section 10 of Dupont, Fliche, and Yang (2020). According to this diagram, neural networks can not offer a good compromise between the complexity of the model (in the sense of understanding its behavior) and its performance. An efficient neural network, from the point of view of the regulator, is often abstruse (the so-called black box phenomenon) and calls for a particularly rigorous model governance.

1.2.2 Hard constraints versus soft constraints

The accuracy of an algorithm is not an end in itself as underlined in the previous subsection. For instance in the case of the interpolation of European option prices, arbitrage constraints have to be imperatively respected in order to meet the obligations of internal bank controls. These arbitrage constraints are declined in shape constraints, i.e. involving the partial derivatives of the interpolator (in particular with respect to the strike or the maturity).

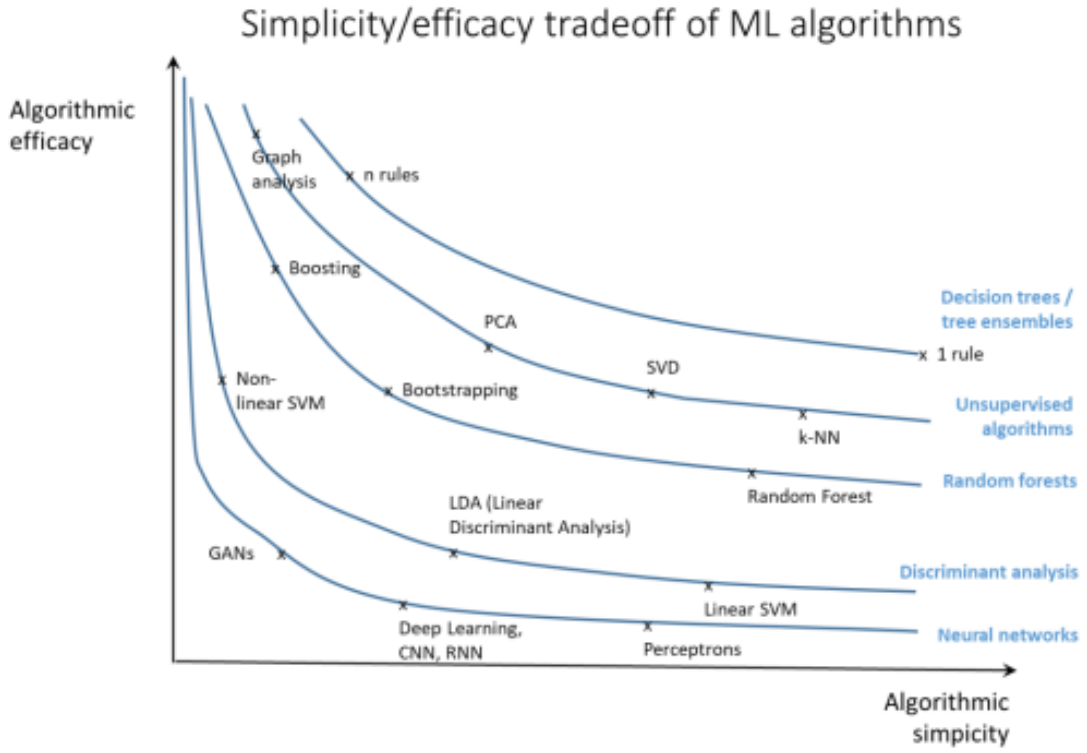


Figure 1.2.1: Compromis simplicité/performance selon la classe d'algorithme d'apprentissage statistique.

A neural network should in theory (see section 6 of Schmidt-Hieber (2020)) see its partial derivatives converge to the sensitivities of price functions. However, obtaining the local minimum of the risk function does not in practice ensure a suitable precision of the neural network derivatives. In addition, the data can itself violate these shape constraints and it is then necessary to regularize the learning of the network so as not to overfit.

There are then two approaches to impose these constraints:

- The hard constraints guarantee the respect of these conditions whatever the input data. This means no-arbitrage, in the interpolation context, whatever the option coordinates.

- Soft constraints penalize the risk function if the no-arbitrage conditions are violated for some observations of the training set. This approach results in penalizations in the optimization routine that can impose non-arbitrability on the training set but offer no guarantee on new observations for example in the testing set.

The hard constraints are reflected in the case of a neural network by the construction of an architecture which integrates these shape constraints (see Dugas, Bengio, Bélisle, Nadeau, and Garcia (2009)). However, the architecture induces a class of functions that is too restrictive with respect to the price function to be approached. The universal approximation theorem is then no longer valid in this case. In addition, there is no architecture ensuring non-arbitrability in cases where the condition is more complex (see in particular the butterfly condition for the interpolation of implied volatility).

Soft constraints, on the other hand, provide the flexibility necessary to penalize complex conditions. The idea of penalization is to ensure compliance with non-arbitrage conditions at the nodes defined by the learning grid. The wager of the flexible constraints is to exploit the neural network regularity so that these conditions are also respected in the neighborhoods of the learning set nodes.

1.2.3 Other non-arbitrable surrogate models

Other statistical learning models can mix interpolation and respect for arbitrage constraints. Among these models we can cite the Gaussian processes which assume that all the prices are distributed according to a Gaussian vector. The option locations, through variables such as maturity or strike, is involved in the calculation of the correlation of different prices. It is the purpose of the kernel which assigns a stronger correlation to prices whose coordinates are close. A Gaussian calculation (see subsection 15.2.1 of Murphy (2012b)) then makes it possible to interpolate the price of new options by exploiting their correlation with the observations of a training grid. Cousin, Maatouk, and Rullière (2016a) propose a method to build Gaussian processes that ensure linear constraints in a hard way. The idea is to simulate the trajectories of a truncated Gaussian process, and more precisely to reject samples of the process which do not respect, for example, monotonicity constraints on a discrete grid. Other linear (equality) constraints are modeled by tuning the distribution of the Gaussian process (see page 14 of Cousin, Maatouk, and Rullière (2016a)).

Aubin-Frankowski and Szabo (2020) model shape constraints for reproducing kernel Hilbert space based estimators. The kernel k in this case plays the role of a change of variable which is supposed to facilitate the learning of an estimator f. The authors reinforce their shape constraints of the type $Df(x_m) + b \geq 0$ to apply them to any node of the training set by adding a constant to obtain $Df(x_m) + b \geq \eta$. This constant η is calibrated according to the regularity of the kernel so as to guarantee the respect of the shape constraints for a neighborhood of all the points x_m of the training set, typically:

$$\eta = \sup_{m\in\{1,..,M\},u\in\mathcal{B}_{\|.\|}(0,1)} \|Dk(x_m,.) - Dk(x_m + \delta u,.)\|$$

However, this procedure is not compatible with non-linear constraints.

1.3 Dealing with empirical data

1.3.1 Intraday data shortcomings

Various data providers (Bloomberg, ICAP, etc.) continuously inform market operators of the valuations of various financial assets or products. But these data cannot be processed in that condition and require additional post-processing. If this data is intraday (i.e. coming from the current market session) then it is very likely that the information is incomplete. Some financial contracts, for example, are less liquid and are not traded during the day or only in a negligible volume.

In addition, the values provided by the data supplier may be inconsistent or appear erroneous for an expert eye. This can be explained by operational reasons (human error, low volume of exchange), or because of temporally inconsistent observations (some data has not been updated). We propose in the chapter 3 a way to correct these data flows.

The data that we will process have in common that they are structured in the form of a tensor. By tensor is meant here a data structure indexed according to several dimensions. For example, the implied volatility of European options is organized as a tensor of order 2 (a matrix) and indexed along to the maturity and the strike. Indexing plays an important role since it defines the notion of proximity between the elements of the tensor. In this report we will only deal with tensors of order 2 at most but it is quite possible to observe tensors of higher order: the implicit volatility of swaptions are for instance arranged according to a cube. In addition, indexation may vary over time in the sense that the maturities observed one day may be different the next day in value and number. We cannot thus be satisfied with identifying an element according to its rank of indices.

1.3.2 Outlier detection

The definition of an outlier is controversial since its detection requires expert advice. Referring to Hawkins (1980), an outlier is an observation that differs significantly from other observations in the same dataset to raise a doubt about its validity. An outlier can be materialized by corrupted, incomplete or even atypical data in terms of shape.

There are several approaches to formalize the detection of abnormal observations, including:

- An outlier is an observation significantly distant from the others according to a metric calculated from the sample features. A clustering analysis, based

for example on the method of K nearest neighbors (Knorr and Ng 1998), can then be applied to the dataset by basing the decision rule on the distance previously mentioned. One way to build this distance is to perform a coordinate change through a PCA (see section 3.3 of (Aggarwal 2017)) or using a kernel (see section 3.4.3 of (Aggarwal 2017)).

- An outlier can be modeled according to a statistical model from which a notion of likelihood derives. An atypical observation then results in a low level of likelihood. Among the statistical models we can cite the Kalman filter (cf. (Ting, Theodorou, and Schaal 2007), (Liu, Shah, and Jiang 2004)) or hidden Markov models (section 9.3.3 of (Aggarwal 2017)).

- An outlier is an observation that does not respect the latent structure of the sample. In this perspective, a low-dimensional representation (with a PCA or an autoencoder according to Aggarwal (2017)) enables us to discriminate an abnormal observation with its reconstruction error.

In chapter 3 our detection of outliers follows the third approach. However, PCA/autoencoders are difficult to reconcile with indexations that vary over time since these projections assume a dimension of the observations that is fixed over time.

1.3.3 Completing observations with variable indexation

Matrix completion is a subcategory of the statistics branch aiming to impute missing values. Completion assumes that the incomplete dataset has a low rank structure, which is justified if the variables in our dataset are strongly related or if the observations are structured around categories. In this part, we will denote by $D \in \mathbb{R}^{m \times n}$ a matrix representing the dataset with m observations (individuals) and n variables. The low rank hypothesis means that we can factorize by singular value decomposition (truncated) the matrix D in the form $U\Sigma V$ with $U \in \mathbb{R}^{m \times r}$, $\Sigma \in \mathbb{R}^{r \times r}$ a diagonal matrix and $V \in \mathbb{R}^{r \times n}$. The lower the rank r and the stronger the connections between the variables/observations, the better the imputation of the missing values will be. If we introduce missing values to D then a SVD is no longer possible and it is then necessary to estimate (U, V, Σ) with the remaining values of the dataset. There are then several methods to estimate (U, V, Σ) such as the softImpute algorithm (Mazumder, Hastie, and Tibshirani 2010) or Alternating Least Square (Hastie, Mazumder, Lee, and Zadeh 2015).

This whole branch of statistical learning is driven by the recommendation systems of digital companies and online marketing. The observations in the dataset represent users of a platform and the features are ratings of sold products. Completion then allows the platform owner to infer the missing ratings in order to propose their customers products that would interest them. This application case

gave rise to a competition (the netflix price) whose dataset now serves as a reference dataset in the literature (cf. (Nguyen, Kim, and Shim 2019)). Bibliographic reviews are available on the subject such as Li, Huang, So, and Zhao (2019) and Nguyen, Kim, and Shim (2019).

However, the literature in completion does not apply to the implicit option volatility data studied in the 3 chapter:

- The columns are not identified with variables since an option can expire one day and leave its place in the column to an option having another residual maturity.
- The matrix structure does not encode information about the indexing of variables. Yet, this information is necessary to define a notion of proximity between the variables.
- Only the last line, corresponding to the current quotation day, contains missing values. As the other lines have been observed in the past, the number of missing values is reduced and localized unlike user experience datasets (a la " netflix").

The 3 chapter aims at reconciling the imputation of missing values with the management of data whose indexing varies from day to day. A low rank structure will always be learned from the past by learning a decoder but the encoder will be implicit to deal with the variable number of entries. Finally, we will only consider connections between the variables (implicit volatilities) and not between the trading days. This is similar in the literature to an item-based approach.

1.4 XVAs compression

1.4.1 The 2008 turmoils

The acronym XVA designates a family of counterparty risk adjustments and more literally the X designates a blank letter preceding Value Adjustments. These risk premiums will be classified into three main categories: default risk premiums, premiums denoting the cost of funding collateral and premiums for financing regulatory capital.

These adjustments aim in part to model two types of losses:

- Depreciation on the market of a financial contract that is only justified by a loss of confidence in the solvency of the counterparty. The loss in value is therefore not linked to the intrinsic characteristics of the contract.

- The default of the counterparty who fails to honor its payments. Contract payments maturing after the default date are then modeled as payoffs to be hedged as part of counterparty risk.

The 2008 financial crisis materialized the losses caused by the counterparty risk. First of all, the number of defaults has increased, in particular in the US mortgage loan sector, impacting the financial markets through securitization. A contagion effect causes credit spreads to rise in all financial markets and a downward pressure on derivatives for fear of numerous bankruptcies. In addition, bank financing is weakened: collaterals are only accepted in return for larger haircuts, interbank rates soar with a spread of 3.5 % against the OIS [3]. With the value of banks' derivative portfolios depreciated and the need for more fundings to face risks, banks must sell some of their assets (stocks, bonds) to restore their capital ratio and confidence. However, these firesales accelerate the depreciation of assets on the financial markets, thus creating a negative spiral. This crisis thus has all the characteristics to which the XVAs wish to respond: valuing the losses linked to defaults, anticipating the costs of financing collateral [4], reinforcing the shareholders' equity of the banks to pass the financial stress periods smoothly.

To formalize these concepts it is necessary to differentiate between the intrinsic value [5] of a contract U and its market value V supposed to capture confidence in the counterparty. The difference $U - V$ values this counterparty risk and refers to the CVA and the FVA which will be described later. Banks therefore list these quantities in their balance sheet so as to refine the valuation of their products and set up hedging strategies.

In this section, we will briefly present, for each counterparty adjustment, the calculation methodologies proposed by the regulator. Then we will discuss a more quantitative formulation of these counterparty adjustments in order to highlight the market incompleteness behind each XVA.

1.4.2 Pricing default risk

The CVA, or more precisely Credit Value Adjustment in English, is a risk premium for the default of the counterparty. In fact, the bank records a loss if the counterparty engaged under a contract with the bank defaults. This loss is measured by

[3]The overnight indexed swap (or overnight swap rate) is considered as a risk-free rate proxy. The LIBOR-OIS spread is a misnomer which refers to the gap between the swap rate based on LIBOR rate and the swap rate based on the overnight rate. Usually the LIBOR (interbank rate) was close to the OIS but the subprime crisis negated this assumption.

[4]Financial asset given as a guarantee.

[5]The intrinsic value (or clean value) depends only on the characteristics of the contract but does not refer to the identity of the signatories.

the client's exposure to the bank if it is negative (i.e. the counterparty owes the bank money).

There are several approaches to assess a so-called accounting CVA, that is to say the CVA that most faithfully reflects the losses caused by the default risk. We will limit ourselves to the unilateral case, that is to say the situation where only the default of the counterparty is valued.

The first approach, called semi-replication (Burgard and Kjaer 2011), is more suited for the resolution by EDP. The term semi refers to the imperfect nature of the hedge: the recovery process R [6] is unknown and there are no hedging instruments on the market allowing a perfect replication of the counterparty's default. Other methodologies have been proposed to value XVAs. We can notably cite Albanese, Crépey, and Chataigner (2018) which models the bank's complete balance sheet. This makes it possible to calculate the FVA for a group of counterparties and to benefit from mutual funding of the collateral. In addition, the KVA is no longer considered as a liability of the bank but is integrated into its equity, which means that the KVA does not intervene in the P&L associated to the counterparty.

The formula for CVA below is demonstrated in section 9.4.1 of Green (2015) in this semi-replication setting. In chapter 4, the optimized metric will be calculated according to this formula (1.1).

$$\text{CVA} = (1 - R) \int_0^T \lambda_{C\,s} e^{-\int_s^t r_u + \lambda_{C\,u} du} \mathbb{E}\left[(V_s - X_s)^+\right] ds \tag{1.1}$$

with X_s the amount of collateral posted on the date s, T the maturity of the portfolio and λ_C the credit spread of the counterpart.

In reality this formula is the result of a simplistic assumption that assumes the independence between the exposure $V - X$ and the default risk of the counterparty, measured by $lambda_C$. Thus in Albanese, Crépey, and Chataigner (2018), the default risk should be modeled more generally according to the following formula:

$$\text{CVA} = (1 - R)\mathbb{E}\left[(V_\tau - X_\tau)^+ \mathbb{1}_{\tau \leq T}\right] \tag{1.2}$$

where tau designates the counterpart default time.

It is interesting to dwell on the operational execution of this kind of formula. The clean value V can be recorded as a tensor of order 3 called mark-to-market cube. It contains all trade values for all exposure trajectories, all dates until maturity. The challenge for the bank's credit desk is to value a mark-to-market cube that is consistent with the trading desk's pricing methodology. This trading desk, which values and hedges trades independently of the creditworthiness, must fit in with the credit desk on the environment it uses (market data, calibration of model parameters) in order to match the counterparty risk hedging strategy of the credit

[6] Percentage of the exposure lost at the time of default.

desk with market risk hedge of the trading desk. Indeed, exposure plays a central role in the valuation of XVAs and affects their sensitivities. In the case of CVAs, the exposure affects the hedge against non-credit instruments such as yield curves for example. This rigorous and automated exchange of information is referred in chapter 4 under the name of desk segregation. It is moreover implicitly encouraged by the regulator which, in its revision of the CVA calculations (see section 4 of Basel Committee on Banking Supervision (2015)) or with FRTB-CVA, asks to include the hedge in the calculations of capital charges. In addition, regulatory CVAs must use parameters (drifts, probabilities of default, etc.) calibrated according to the risk-neutral approach (page 2-3 of Basel Committee on Banking Supervision (2015)). The regulator therefore tends to align its regulatory calculations with accounting valuations (IFRS 13 still according to page 2 of the same report) which are aligned with the valuations of the trading desk.

Finally, the complexity of the Mark-to-market cube is very important and the CVA is not linear with respect to the portfolio. If the bank concludes a new trade with the counterparty then the mark-to-market cube must be revalued in its entirety to know the new amount of the CVA. This setback will be limited in chapter 4 thanks to the incremental computation.

1.4.3 Pricing collateral funding costs

We justified in the previous sub-section that a perfect replication linked to the default of the counterpart is impossible. Therefore, exposure must be minimized in order to reduce the residual risk. Standard market practice use collateral as a pledge of the value of the portfolio. If the bank's exposure is negative then it must post collateral and otherwise it is up to the counterpart to post the collateral to the bank. This exchange of collateral also occurs between the bank and the entities with which the bank hedges (back-to-back hedge) except that the bank posts collateral with these entities if it receives collateral from the counterpart.

Generally the collateral is a good quality financial security such as a sovereign bond of good rating or simply cash. If the guarantee is deemed unreliable then a discount (haircut) is applied to the value of the collateral and it is necessary to mobilize a larger notional amount as a pledge.

There are several categories of collateral characterized by their method of calculation:

- The Variation margin which is indexed to the amount of MtM at each refresh date. The amount posted between two margin calls depends on the amount of collateral already posted and various clauses such as minimum transfer amounts, trigger thresholds (see chapter 6 of (Gregory 2015) for more details) ... The variation margin can be reused as collateral in other

transactions by the collateral recipient before the transaction expires. This means that the cost of borrowing collateral for the hedge is zero if we can post the collateral provided by the counterpart. On the other hand, if the counterparty does not post collateral, because it has not signed an credit support annex, then the bank must borrow this collateral on the markets at a cost generally greater than its remuneration (often equal to the OIS) . The Funding Value Adjustment (FVA, for its formulation see chapter 9 of (Green 2015)) corresponds to the cost of financing the variation margin if the counterpart does not or not enough.

- The initial margin is a minimum safety cushion aimed at covering the risk of the MtM slipping between two margin calls [7]. For each margin call, the amount of collateral is based on a portfolio risk measure: Value-at-Risk or a derivative of VaR based on sensitivities in the case of an IM SIMM (Standardized initial margin method). The cost of financing this initial margin gives rise to the Margin Value Adjustment (or MVA, see chapter 16 of (Gregory 2015) or the appendix of (Crépey, Hoskinson, and Saadeddine 2021) for its formulation). The initial margin cannot in general be reused by its recipient for other transactions, that is to say it is segregated.

Funding costs occur on other occasions such as clearing houses (see (Armenti and Crépey 2017)). The regulator also encourages the clearing of derivative transactions or even makes it mandatory (see section 9.3 of (Gregory 2015)). CCPs will not be included in the study of chapter 4.

b Collateralisation can reduce or even cancel CVA and FVA. However, it can give rise to other costs such as MVA.

1.4.4 Pricing capital funding costs

Along with these accounting entries, the regulator wanted to impose capital reserves on banks (the first pillar of the Basel regulation) in order to prevent the appearance of Ponzi schemes. The principle of this financial arrangement is to enter into new contracts in order to finance the losses of a pre-existing portfolio. However, capital reserves are expensive to raise from shareholders since they generally require a much higher return than the funding cost on the interbank market, for example. Setting up a trade is then more expensive and prevents the construction of a Ponzi scheme. Regulatory capital reserves are primarily conservative and are not intended to exactly cover a loss but rather to provide a safety cushion in anticipation of turmoils.

Among the capital reserves we can mention:

[7] This slippage risk is valued under the term Margin period of risk.

- The counterparty credit risk capital charge that covers the default of the bank's customers.
- The CVA capital charge which provisions for depreciations of the value of the bank's assets if the creditworthiness of its debtors deteriorates.
- The Risk weighted asset which serves as a reference for provisioning the bank's regulatory capital (Basel ratio). All the bank's balance sheet and all risk classes are captured by the RWA.

Details of the formulas for each capital reserve can be found in (Basel Committee on Banking Supervision 2015) (or chapter 8 of (Gregory 2015)) but we can identify some similarities. First of all, the regulator offers at least two calculation methodologies to adapt to the computation capacities of each establishment. Among these approaches there is always a parametric method, simpler to calculate, but more conservative. Another, more complex approach allows the use of internal bank valuation models, which often requires simulating exposure as for other XVAs.

In addition, all these quantities designate amounts of capital to be reserved but it is their funding cost which is charged to the customer. The capital value adjustment (KVA) aims to estimate this cost and then add it to the clean value V of the contract alongside the CVA and the FVA. However, the capital reserve involved in the KVA is not necessarily based on one of the previous regulatory capital reserves but can alternatively be based on economic capital. Economic capital is a measure of risk of how much the bank must have in reserve to weather a crisis. This is generally a Value-at-Risk or an expected shortfall calculated on the bank's losses with an one year horizon. Noting h the hurdel rate of shareholders on capital invested in the bank, Crépey, Hoskinson, and Saadeddine (2021) in their appendix formulate the KVA as:

$$KVA = \mathbb{E}\left[\int_0^{T\wedge\tau} he^{-hs}\max\left(EC_s, KVA_s\right)ds\right] \tag{1.3}$$

The KVA thus captures all losses arising from counterparty risk that cannot be hedged. Like the CVA, the KVA and other capital reserves could be optimized. However, the computational complexity of the KVA makes its compression inaccessible for the moment. To our knowledge, only the MVA (in a simplistic form) with (Kondratyev and Giorgidze 2017) has been the subject of a compression procedure in the literature. We propose to optimize the CVA and to describe the operational stakes in the chapter 4.

Chapter 2

Arbitrage-Free neural network

Machine learning for option pricing has emerged as a novel methodology for fast computations with applications in calibration and computation of Greeks. However, most of these approaches do not enforce any no-arbitrage conditions. In this article, we develop a neural network approach for no-arbitrage interpolation of European vanilla option prices. In particular, we demonstrate the modification of the standard deep learning methodology to enforce the no-arbitrage constraint and we specify the experimental design parameters that are needed for adequate performance. A novel component is the use of the Dupire formula to enforce bounds on the local volatility associated with (non-arbitrable) option prices, during the network fitting. Numerical results[1] on real datasets of DAX and SPX vanilla options demonstrate the numerical error in the price, implied volatility and local volatility surface.

2.1 Introduction

A recent approach to option pricing involves reformulating the pricing problem as a surrogate modeling problem. Once reformulated, the problem is amenable to standard supervised machine learning methods such as Neural networks and Gaussian processes. This is particularly suitable in situations with a need for fast computations and a tolerance to in-exact approximation.

In their seminal paper, (Hutchinson, Lo, and Poggio 1994) use a radial basis function neural network for delta-hedging. Their network is trained to Black-Scholes prices, using the time-to-maturity and the moneyness as input variables,

[1] A Python notebook, compatible with Google Colab, and accompanying data are available in https://github.com/mChataign/Beyond-Surrogate-Modeling-Learning-the-Local-Volatility-Via-Shape-Constraints. Due to file size constraints, the notebook must be run to reproduce the figures and results in this article.

without shape constraints. They use the hedging performance of the ensuing delta-hedging strategy as a performance criterion. (Garcia and Gençay 2000) and (Gençay and Qi 2001) improve the stability of the previous approach by adding the outputs of two such neural networks, weighted by respective moneyness and time-to-maturity functionals. (Dugas, Bengio, Bélisle, Nadeau, and Garcia 2009) introduce the first neural network architecture guaranteeing arbitrage-free vanilla option pricing. However, (Ackerer, Tagasovska, and Vatter 2019) show that the corresponding hard constrained networks are very difficult to train in practice. Instead, they advocate the learning of the implied volatility (rather than the prices) by a standard feedforward neural network with soft-constraints, i.e. regularization, which penalizes calendar spread and butterfly arbitrages. The call and put prices must also be decreasing and increasing by strike respectively.

We propose simple neural network architectures and training methodology which satisfy these shape constraints. In contrast to Dugas, Bengio, Bélisle, Nadeau, and Garcia (2009), following Ackerer, Tagasovska, and Vatter (2019), we also explore soft constraints alternatives to hard constraints in the network configuration, due to the loss of representation power of the latter. Moreover, a novel aspect of our approach is to focus on the associated local volatility surface, considered both for itself and as a penalization device in our soft constraints approach.

Another contribution of the chapter is to introduce a neural network approximation of the implied volatility surface, again penalizing arbitrages on the basis of the Dupire formula, which is also used for extracting the corresponding local volatility surface. This is all evidenced on an SPX option dataset and benchmarked against the SSVI industry standard and an arbitrage free GP interpolation approach.

This section is the merge of two different articles (see (Chataigner, Crépey, and Dixon 2020) and (Chataigner, Cousin, Crépey, Dixon, and Gueye 2021)) using different datasets. As a result training methodology and neural architecture evolved accordingly to each dataset. During all of our experiments, neural local volatility is derived though back-propagation algorithm and then is backtested against various benchmarks depending on the dataset.

2.2 Problem Statement

We consider European vanilla option prices on a stock or index S. We assume that a deterministic short interest rate term structure $r(t)$ of the corresponding economy has been bootstrapped from the its zero coupon curve, and that a term structure of deterministic continuous-dividend-yields $q(t)$ on S has then been extracted from the prices of the forward contracts on S. The assumption of deterministic rates

and dividends is for consistency with local volatility models, in the perspective, later in the chapter, of numerical experiments on equity options (using (Crépey 2002) as a benchmark).

Without restriction given the call-put parity relationship, we only consider put option prices. We denote by $P^\star(T,K)$ the market price of the put option with maturity T and strike K on S, observed for a finite number of pairs (T,K) at a given day.

Our goal is to construct a nonarbitrable put price surface $P : \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}_+$ in $\mathcal{C}^{1,2}((0,+\infty)\times\mathbb{R}_+^\star)\cap\mathcal{C}^0(\mathbb{R}_+\times\mathbb{R}_+^\star)$, interpolating $P^\star$ up to some error term. As is well known, the corresponding local volatility surface, say $\sigma(\cdot,\cdot)$, is given by the Dupire (1994) formula

$$\frac{\sigma^2(T,K)}{2} = \frac{\partial_T P(T,K) + (r-q)K\partial_K P(T,K) + qP(T,K)}{K^2\partial^2_{K^2}P(T,K)}.$$

In terms of $P(T,k) = \exp\left(\int_0^T q(t)dt\right)P(T,K)$, where $k = K\exp\left(-\int_0^T (r(t)-q(t))dt\right)$, the formula reads

$$\frac{\sigma^2(T,K)}{2} = dup(T,k) := \frac{\partial_T P(T,k)}{k^2\partial^2_{k^2}P(T,k)} =: \frac{\mathrm{cal}_T(P)}{\mathrm{butt}_k(P)}(T,k). \tag{2.1}$$

We then learn the modified market prices $P^\star = P^\star(T,k)$. Given a rectangular domain of interest in maturity and strike, we rescale further the inputs, $T' = (T - min(T))/(max(T) - min(T))$ and $k' = (k - min(k))/(max(k) - min(k))$, so that the domain of the pricing map is $\Omega := [0,1]^2$. This rescaling avoids any one independent variable dominating over another during the fitting. For ease of notation, we shall hereon drop the $'$ superscript.

For the Dupire formula (2.1) to be meaningful, its output must be nonnegative. This holds, in particular, whenever the interpolating map P exhibits nonnegative derivatives w.r.t. T and second derivative w.r.t. k, i.e.

$$\partial_T P(T,k) \geq 0,\ \partial^2_{k^2}P(T,k) \geq 0. \tag{2.2}$$

In both networks considered in the chapter, these derivatives are available analytically via the neural network automatic differentiation capability. Hard or soft constraints can be used to enforce these shape properties, exactly in the case of hard constraints and approximately (via regularization) in the case of soft constraints. More generally, see (Roper 2010, Theorem 2.1) for a full and detailed statement of the static non-arbitrage relationships conditions on European vanilla call (easily transposable to put) option prices, also including, in particular, an initial condition at $T = 0$ given by the option payoffs. During the DAX experiments, this initial payoff condition will also be incorporated to our learning schemes, in a way described in Section 2.4.1.

If we now consider implied volatilities Σ instead of price options P then our neural network will take as input maturity T and log-forward maturity $\kappa = \log(\frac{k}{S_0})$. The corresponding local volatility surface σ is then given by the following local volatility implied variance formula, i.e. the Dupire formula stated in terms of the implied total variance[2] $\Theta(T,\kappa) = \Sigma^2(T,\kappa)T$ (assuming Θ of class $\mathcal{C}^{1,2}$ on $\{T > 0\}$):

$$\sigma^2(T,K) = \frac{\partial_T \Theta}{1 - \frac{\kappa}{\Theta}\partial_\kappa \Theta + \frac{1}{4}\left(-\frac{1}{4} - \frac{1}{\Theta} + \frac{\kappa^2}{\Theta^2}\right)(\partial_\kappa \Theta)^2 + \frac{1}{2}\partial_{\kappa^2}\Theta}(T,\kappa) =: \frac{\mathrm{cal}_T(\Theta)}{\mathrm{butt}_\kappa(\Theta)}(T,\kappa). \tag{2.3}$$

T and κ are subject to the same scalings as T and k in the context of neural network learning prices.

Similarly to price based neural networks, no arbitrage conditions in presence of implied volatilities are also obtained through back-propagation algorithm. In the rest of this chapter we refer to butterfly constraint as the condition with respect to strike-logmoneyness and calendar condition denotes the condition with respect to maturity, The following table summarizes the above no arbitrage condition in any experimental setting.

Learned quantity \ Condition	calendar	butterfly
Price	$\partial_T P \geq 0$	$k^2 \partial_{k^2}^2 P(T,k) \geq 0$
Implied Volatility	$\partial_T \Theta \geq 0$	$\partial_T \Theta 1 - \frac{\kappa}{\Theta}\partial_\kappa \Theta + \frac{1}{4}\left(-\frac{1}{4} - \frac{1}{\Theta} + \frac{\kappa^2}{\Theta^2}\right)(\partial_\kappa \Theta)^2 + \frac{1}{2}\partial_{\kappa^2}\Theta(T,\kappa) \geq 0$

2.3 Shape Preserving Neural Networks

We consider parameterized maps $p = p_{\mathbf{W},\mathbf{b}}$

$$(T,k) \ni \mathbb{R}^2_+ \xrightarrow{P} P_{\mathbf{W},\mathbf{b}}(T,k) \in \mathbb{R}_+,$$

given as deep neural networks with two hidden layers. As detailed in (Goodfellow, Bengio, and Courville 2016), these take the form of a composition of simpler functions:

$$P_{\mathbf{W},\mathbf{b}}(x) = f^{(3)}_{W^{(3)},b^{(3)}} \circ f^{(2)}_{W^{(2)},b^{(2)}} \circ f^{(1)}_{W^{(1)},b^{(1)}}(x),$$

where

$$\mathbf{W} = (W^{(1)}, W^{(2)}, W^{(3)}) \text{ and } \mathbf{b} = (b^{(1)}, b^{(2)}, b^{(3)})$$

are weight matrices and bias vectors, and the $f^{(l)} := \varsigma^{(l)}(W^{(l)}x + b^{(l)})$ are semi-affine, for nondecreasing activation functions $\varsigma^{(l)}$ applied to their (vector-valued) argument componentwise. Any weight matrix $W^{(\ell)} \in \mathbb{R}^{m \times n}$ can be expressed as

[2]This follows from the Dupire formula by simple transforms detailed in (Gatheral 2011, p.13).

an n column $W^{(\ell)} = [\mathbf{w}_1^{(\ell)}, \ldots, \mathbf{w}_n^{(\ell)}]$ of m-vectors, for successively chained pairs (n, m) of dimensions varying with $l = 1, 2, 3$, starting from $n = 2$, the number of inputs, for $l = 1$, and ending up with $m = 1$, the number of outputs, for $l = 3$.

A feedforward NN with weights $\mathbf{W}$, biases $\mathbf{b}$ and smooth activation functions is also used for parameterizing the implied volatility and total variance, which we denote by

$$\Sigma = \Sigma_{\mathbf{W},\mathbf{b}},\ \Theta = \Theta_{\mathbf{W},\mathbf{b}}.$$

2.3.1 Hard Constraints Approach

In the hard constraints case, our network is sparsely connected in the sense that, with $x = (T, k)$ as above,

$$f^{(1)}_{W^{(1)},b^{(1)}}(x) = [f^{(1,T)}_{W^{(1,T)},b^{(1,T)}}(T), f^{(1,k)}_{W^{(1,k)},b^{(1,k)}}(k)],$$

where $W^{(1,T)}, b^{(1,T)}$ and $W^{(1,k)}, b^{(1,k)}$ correspond to parameters of sub-graphs for each input, and

$$f^{(1,T)}(T) := \varsigma^{(1,T)}(W^{(1,T)}T + b^{(1,T)}),\ f^{(1,k)}(k) := \varsigma^{(1,k)}(W^{(1,k)}k + b^{(1,k)}).$$

To impose the shape constraints relevant for put options, it is then enough to restrict ourselves to nonnegative weights, and to convex (and nondecreasing) activation functions, namely

$$\text{softplus}(x) := \ln(1 + e^x),$$

except for $\varsigma^{(1,T)}$, which will be taken as an S-shaped sigmoid $(1+e^{-x})^{-1}$. Imposing non-negative constraints on weights can be achieved in back-propagation using projection functions applied to each weight after each gradient update.

Hence, the network is convex and nondecreasing in k, as a composition (restricted to the k variable) of convex and nondecreasing functions of k. In T, the network is nondecreasing, but not necessarily convex, because the activation function for the maturity subnetwork hidden layer is not required to be convex - in fact, we choose a sigmoid function.

However such hard constrained architecture is not possible when dealing with implied volatility. The butterfly condition $\text{butt}_\kappa(\Theta) \geq 0$ does no longer reduce with respect to κ and we can only employ soft constraint as described in 2.3.2.

Figure 2.3.1 illustrates the shape preserving feed forward architecture with two hidden layers containing 10 hidden nodes. For avoidance of doubt, the figure is not representative of the number of hidden neurons used in our experiments. However,

the connectivity is representative. The first input variable, T, is only connected to the first 5 hidden nodes and the second input variable, k, is only connected to the last 5 hidden nodes. Effectively, two sub-networks have been created where no information from the input layer crosses the sub-networks until the second hidden layer. In other words, each sub-network is a function of only one input variable. This property is the key to imposing different hard shape constraints w.r.t. each input variable.

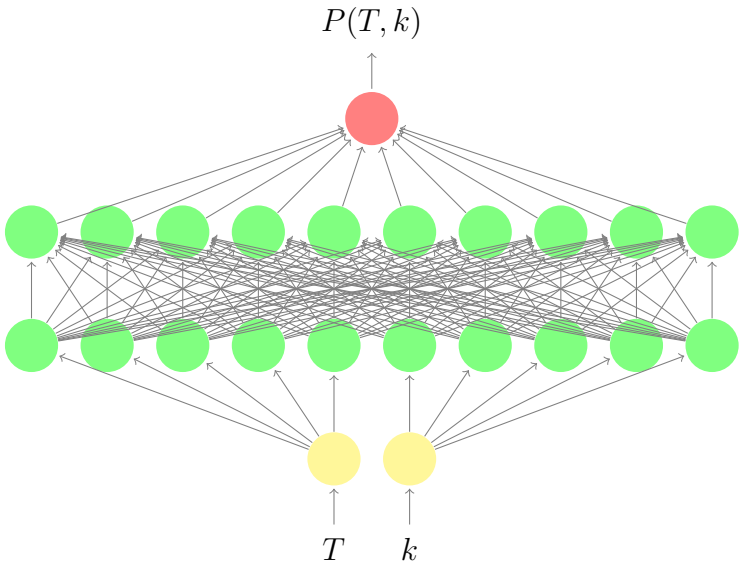


Figure 2.3.1: *A shape preserving (sparse) feed forward architecture with one hidden layer containing 10 hidden nodes. The first input variable, T, is only connected to the 5 left most hidden nodes and the second input variable, k, is only connected to the 5 right most hidden nodes.*

2.3.2 Soft Constraints Approach

However, Theorem 4.1 in (Ohn and Kim 2019), provides support for our observation, presented in Section 2.4.2, that sparsening the network (i.e. splitting) increases the approximation error. Hence, in what follows, we also consider the so called soft constraints approach using a fully connected network, where the static no arbitrage conditions (2.2) are favored by penalization, as opposed to imposed to hold exactly in the previous hard constraint approach.

Note that only the "hard constraints" approach theoretically guarantees that the associated Dupire formula (2.1) returns a positive function. While soft constraints reduce the risk of static arbitrage in the sense of mismatch between model

and market prices, they do not however fully prevent arbitrages in the sense of violations of the shape conditions (2.2) in the predicted price surface, especially far from the grid nodes of the training set.

In particular, the penalties only control the corresponding derivatives on a discrete set of points. Compliance with the no-arbitrage constraints on the majority of the points in the test set is due only to the regularity of these derivatives. This is not a novel idea. (Aubin-Frankowski and Szabo 2020) exploit reproducing kernel Hilbert space regularity to ensure conditions on derivatives in a hard constraint manner with a second order cone optimization. They add a margin to the penalizations so that these derivative conditions are ensured over a targeted neighbourhood of training points. In our case we do not add such a margin to our penalizations. Instead, we add a further half-variance bounds penalization, which both favors even more the shape constraints (without guaranteeing them in theory) and stabilizes the local volatility function calibration, as detailed below. This penalty is activated when the half-variance (see formula 2.1) leaves an interval especially if it becomes negative.

In addition we propose a novel approach on the SPX dataset which enables the network to emancipate from training options coordinates (that is, from the specific (T_i, k_i) or (T_i, κ_i) of the dataset). This is realized with the introduction of a penalization grid which defines a mesh of points where soft constraints are assessed. If that grid is refined enough then we can have a degree of trust on the entire domain (Ω) concerning the respect of arbitrage condition thanks to neural network regularity. This additional mesh is flexible and allows arbitrage-free extrapolation.

2.3.3 Learning problems

In general, to fit our fully connected or sparse networks to the available option market prices at a given time, we solve a loss minimization problem of the following form (with $\lambda = 0$ in the non-penalized and hard-constrained cases), using observations $\{x_i = (T_i, k_i), P^\star(x_i)\}_{i=1}^n$ of n maturity-strike pairs and the corresponding market put prices:

$$\min_{\mathbf{W},\mathbf{b}} \ \frac{1}{n}\sum_{i=1}^{n}\Big(|P^\star(x_i) - P(x_i)| + \lambda^\mathsf{T}\phi(x_i)\Big). \tag{2.4}$$

Here $p = P_{\mathbf{W},\mathbf{b}}$, $\phi = \phi_{\mathbf{W},\mathbf{b}}$ is a regularization penalty vector

$$\phi := [(\partial_T P)^-, (\partial^2_{k^2} P)^-, (dup - \overline{a})^+ + (dup - \underline{a})^-],$$

and dup is related to P through (2.1). The choice to measure the error $P^\star - P$ under the L_1 norm, rather than L_2 norm, in (2.4) is motivated by a need to

avoid allocating too much weight to the deepest in-the-money options. Note that (Ackerer, Tagasovska, and Vatter 2019) consider a combination of L_1 and L_2 norms. In a separate experiment, not reported here, we additionally investigated using the market convention of vega weighted option prices, albeit to no effect beyond simply using L_1 regularization.

The loss function is non-convex, possessing many local minima and it is generally difficult to find a global minimum. The first two elements in the penalty vector favor the shape conditions (2.2) and the third element favors lower and upper bounds $\underline{a}$ and $\overline{a}$ on the estimated half-variance, *dup*, where constants (which could also be functions of time) $0 < \underline{a} < \overline{a}$ respectively denote desired lower and upper bounds on the surface (at each point in time). Of course, as soon as penalizations are effectively used (i.e. for $\lambda \neq 0$), a further difficulty, typically involving grid search, is the need to determine suitable values of the corresponding "Lagrange multipliers"

$$\lambda = (\lambda_1, \lambda_2, \lambda_3) \in \mathbb{R}^3_+, \tag{2.5}$$

ensuring the right balance between fit to the market prices and the targeted constraints.

Algorithm 1 The NN-Price algorithm for local volatility surface approximation.

Data: Market option price surface P_*

Result: The local volatility surface $\sqrt{2\frac{\mathrm{cal}_T}{\mathrm{butt}_k}(P_{\hat{\mathbf{W}},\hat{\mathbf{b}}})}$

1 $(\hat{\mathbf{W}}, \hat{\mathbf{b}}) \leftarrow$ Minimize the penalized training loss (2.4) w.r.t. $(\mathbf{W}, \mathbf{b})$;

2 $\sqrt{\frac{\mathrm{cal}_T}{\mathrm{butt}_k}(P_{\hat{\mathbf{W}},\hat{\mathbf{b}}})} \leftarrow$ AAD differentiation of the trained NN option price surface

Our second goal is to use neural nets (NN) to construct an implied volatility (IV) put surface $\Sigma : \mathbb{R}_+ \times \mathbb{R} \to \mathbb{R}_+$, interpolating implied volatility market quotes Σ_* up to some error term, both being stated in terms of a put option maturity T and log-(forward) moneyness $\kappa = \log(\frac{k}{S_0}) = \log\left(\frac{K}{S_0}\right) - (r-q)T$. The advantage of using implied volatilities rather than prices, both being in bijection via the Black-Scholes put pricing formula as well known, is their lower variability, hence better performance as we will see.

The terms $\mathrm{cal}_T(\Theta_{\mathbf{W},\mathbf{b}})$ and $\mathrm{butt}_\kappa(\Theta_{\mathbf{W},\mathbf{b}})$ are also available analytically, by automatic differentiation, which we exploit below to penalize calendar spread arbitrages, i.e. negativity of $\mathrm{cal}_T(\Theta)$, and butterfly arbitrage, i.e. negativity of $\mathrm{butt}_\kappa(\Theta)$.

An optimization challenge is that maturity-log moneyness pairs with quoted option prices are unevenly distributed and the NN may favor fitting to a cluster of quotes to the detriment of fitting isolated points. To remedy this non-uniform data fitting problem, we re-weight the SPX observations by the Eu-

clidean distance between neighboring points. More precisely, given n observations $\chi_i = (T_i, \kappa_i)$ of maturity-log moneyness pairs and of the corresponding market implied volatilities $\Sigma_*(\chi_i)$, we construct the $n \times n$ distance matrix with general term $d(\chi_i, \chi_j) = \sqrt{(T_j - T_i)^2 + (\kappa_j - \kappa_i)^2}$. We then define the loss weighting w_i for each point χ_i as the distance $w_i = \min_{j, j \neq i} d(\chi_i, \chi_j)$. with the closest point. These modifications aim at reducing error for any isolated points. In addition, in order to avoid linear saturation of the neural network, we apply a further log-maturity change of variables (adapting the partial derivatives accordingly). We note however that loss presented in the following sections are standard root mean squared error which do not involve any reweighting or log-transform. This allows a fair comparison of all models.

Learning the weights $\mathbf{W}$ and biases $\mathbf{b}$ to the data subject to no arbitrage soft constraints (i.e. with penalization of arbitrages) then takes the form of the following (nonconvex) loss minimization problem:

$$\underset{\mathbf{W},\mathbf{b}}{\arg\min} \quad \sqrt{\frac{1}{n}\sum_i \left(w_i \frac{\Sigma_{\mathbf{W},\mathbf{b}}(\chi_i) - \Sigma_*(\chi_i)}{\Sigma_*(\chi_i)} \right)^2} + \frac{\mathbb{P}_w}{m} \sum_{\xi \in \Omega_h} \lambda^\intercal \mathcal{R}(\Theta_{\mathbf{W},\mathbf{b}})(\xi), \qquad (2.6)$$

where $\lambda = [\lambda_1, \lambda_2, \lambda_3]^\top \in \mathbb{R}^3_+$ and

$$\mathcal{R}(\Theta) = [\mathrm{cal}_T^-(\Theta), \mathrm{butt}_\kappa^-(\Theta), \big(\frac{\mathrm{cal}_T}{\mathrm{butt}_\kappa}(\Theta) - \overline{a}\big)^+ + \big(\frac{\mathrm{cal}_T}{\mathrm{butt}_\kappa}(\Theta) - \underline{a}\big)^-]^\top$$

is a regularization penalty vector evaluated over a penalty grid Ω_h with m nodes as detailed below. The error criterion is calculated as a root mean square error on relative difference, so that it does not discriminate high or low implied volatilities. The first two elements in the penalty vector $\mathcal{R}(\Theta)$ favor the no-arbitrage conditions (2.2) and the third element favors desired lower and upper bounds $0 < \underline{a} < \overline{a}$ (constants or functions of T) on the estimated local variance $\sigma^2(T, K)$. In order to adjust the weight of penalization, we multiply our penalties by the weighting mean $\mathbb{P}_w := \frac{1}{m}\sum_i w_i$.

As with the learning problem 2.4, suitable values of the "Lagrange multipliers" λ, ensuring the right balance between fit to the market implied volatilities and the constraints, are then obtained by grid search. Of course a soft constraint (penalization) approach does not fully prevent arbitrages. However, for large λ, arbitrages are extremely unlikely to occur, except perhaps very far from Ω. With this in mind, we use a penalty grid Ω_h that extends well beyond the domain of the IV interpolation. As previously discussed in section 2.3.2, this is intended so that the penalty term penalizes arbitrages outside of the domain used for IV Interpolation. See Algorithm 3.1 for the pseudo-code of the NN approach.

Algorithm 2 The NN-IV algorithm for local volatility surface approximation.

Data: Market implied volatility surface Σ_*

Result: The local volatility surface $\sqrt{\frac{\text{cal}_T}{\text{butt}_\kappa}(\Theta_{\hat{\mathbf{W}},\hat{\mathbf{b}}})}$

1 $(\hat{\mathbf{W}}, \hat{\mathbf{b}}) \leftarrow$ Minimize the penalized training loss (2.6) w.r.t. $(\mathbf{W}, \mathbf{b})$;

2 $\sqrt{\frac{\text{cal}_T}{\text{butt}_\kappa}(\Theta_{\hat{\mathbf{W}},\hat{\mathbf{b}}})} \leftarrow$ AAD differentiation of the trained NN implied vol. surface

We also want to assess the local volatility surface that arises from the trained neural nets. Local volatility is characterized by below diffusion :

$$\frac{dS_t}{S_t} = (r(t) - q(t))\,dt + \sigma(t, S_t)dW_t, \tag{2.7}$$

Therefore we should retrieve for any dataset the options prices by plugging our neural local volatility in equation 2.7.

This is done with our DAX dataset in section 2.4.4 by Monte Carlo simulation whereas our backtests on SPX dataset include a Monte Carlo simulation and Crank-Nicolson PDE scheme.

2.4 DAX Numerical Experiments

2.4.1 Experimental Design

As a benchmark, reference method for assessing the performance of our neural network approaches, we use the Tikhonov regularization approach surveyed Section 9.1 of (Crépey 2013), i.e. nonlinear least square minimization of the squared distance to market prices plus a penalisation proportional to the H^1 squared norm of the local volatility function over the (time, space) surface (or, equivalently, to the L^2 norm of the gradient of the local volatility). Our motivation for this choice as a benchmark is, first, the theoretical, mathematical justification for this method provided by Theorems 6.2 and 6.3 in Crépey (2003). Second, it is price (as opposed to implied volatility) based, which makes it at par with our focus on *price based* neural network local volatility calibration schemes with DAX options. Third, it is non parametric ('model free' in this sense), like our neural network schemes again, and as opposed to various parameterizations such as SABR or SSVI that are used as standard in various segments of the industry, but come without theoretical justification for robustness, are restricted to specific industry segments on which they play the role of a market consensus, and are all implied volatility based. Fourth, an efficient numerical implementation of the Tikhonov method (as we call it for brevity hereafter), already put to the test of hundreds of real datasets in the context of Crépey (2004), is available through Crépey (2002). Fifth, this method

is itself benchmarked to other (spline interpolation and constrained stochastic control) approaches Section 7 of Crépey (2002).

Our training sets are prepared using daily datasets of DAX index European vanilla options of different available strikes and maturities, listed on the 7^{th}, 8^{th} (by default below), and 9^{th}, August 2001, i.e. same datasets as already used in previous work Crépey (2002, Crépey (2004), for benchmarking purposes. The corresponding values of the underlying are $S = 5752.51, 5614.51$ and 5512.28. The associated interest rate and dividend yield curves are constructed from zero-coupon and forward curves, themselves obtained from quotations of standard fixed income linear instruments and from call/put parity applied to the option market prices. Each training set is composed of about 200 option market prices plus the put payoffs for all strikes present in the training grid. For each day of data (see e.g. Figures 2.4.1-2.4.2), a test set of about 350 points is generated by computing, thanks to a trinomial tree, option prices for a regular grid of strikes and maturities, in the local volatility model calibrated to the corresponding training set by the benchmark Tikhonov calibration method of (Crépey 2002).

Each network has two hidden layers, each with 200 neurons per hidden layer. Note that (Dugas, Bengio, Bélisle, Nadeau, and Garcia 2009) only uses one hidden layer. Two was found important in practice in our case. All networks are fitted with an ADAM optimizer. In order to achieve the convergence of the training procedure toward a local minimum of the loss criterion, the learning rate is divided by 10 whenever no improvement in the error on the training set is observed during 100 consecutive epochs. The total number of epochs is limited to $10,000$ because of the limited number of market prices. Thus we opt for a batch learning with numerous epochs.

After training, a local volatility surface is extracted from the interpolated prices by application of the Dupire formula (2.1), leveraging on the availability of the corresponding exact sensitivities, i.e., using automatic algorithmic differentiation (AAD) and not finite differences. This local volatility surface is then compared with the one obtained in (Crépey 2002).

Moreover, we will assess numerically four different combinations of network architectures and optimization criteria, i.e.

- sparse (i.e. split) network and hard constraints, so $\lambda_1 = \lambda_2 = 0$ in (2.4)-(2.5),
- sparse network but soft constraints, i.e. ignoring the non-negative weight restriction in Section 2.3.1, but using $\lambda_1, \lambda_2 > 0$ in (2.4)-(2.5),
- dense network and soft constraints, i.e. for $\lambda_1, \lambda_2 > 0$ in (2.4)-(2.5),
- dense network and no shape constraints, i.e. $\lambda_1 = \lambda_2 = 0$ in (2.4)-(2.5).

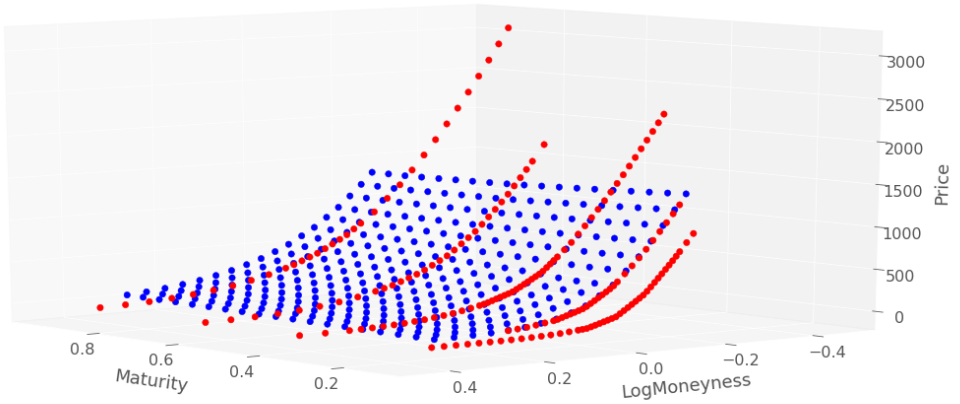


Figure 2.4.1: *DAX put prices from training grid (red points) and testing grid (blue points), 8 Aug 2001.*

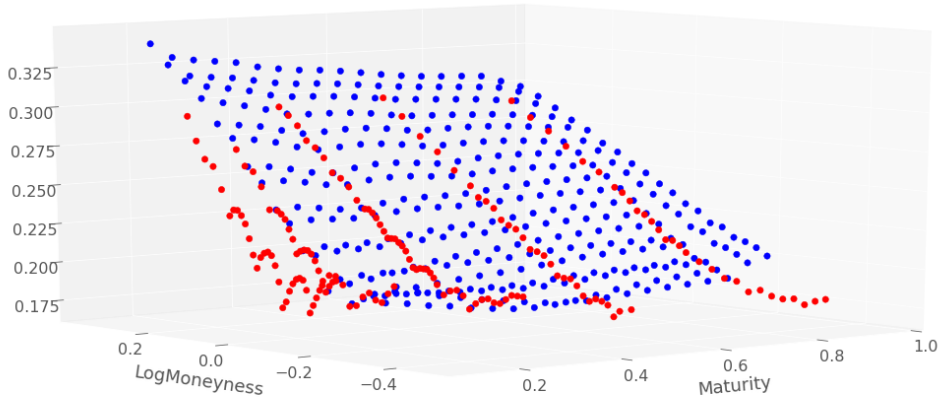


Figure 2.4.2: *Same as Figure 2.4.1 in implied volatility scale.*

Moreover, these four configurations will be tried both without (Section 2.4.2) and with (Section 2.4.3) half-variance bounds penalization, i.e. for $\lambda_3 = 0$ vs. $\lambda_3 > 0$ in (2.4)-(2.5), cases referred to hereafter as without / with Dupire penalization.

In each case the error between the prices of the calibrated model and the market data are evaluated on both the training and an out-of-sample test set. Unless reported otherwise, all numerical results shown below correspond to test sets.

All our numerical experiments were run under google colab with 13 Gos of Ram and a dual core CPU of 2.2GHz.

2.4.2 Numerical Results Without Dupire Penalization

Table 2.4.1 shows the pricing RMSEs for four different combinations of architecture and optimization criteria without half-variance bounds, i.e. for $\lambda_3 = 0$ (2.4)-(2.5). For the sparse network with hard constraints, we thus have $\lambda = 0$. For the sparse and dense networks with soft constraints (i.e. penalization but without the conditions on the weights of Section 2.3), we set $\lambda = [1.0 \times 10^5, 1.0 \times 10^3, 0]$.

The sparse network with hard constraints is observed to exhibit significant pricing error, which suggests that this approach is too limited in practice to approach market prices. This conclusion is consistent with (Ackerer, Tagasovska, and Vatter 2019), who choose a soft-constraints approach in the implied volatility approximation (in contrast to our approach which approximates prices).

	Sparse network		Dense network	
	Hard constraints	Soft constraints	Soft constraints	No constraints
Training dataset	28.13	6.87	2.28	2.56
Testing dataset	28.91	4.09	3.53	3.77
Indicative training times	200s	400s	200s	120s

Table 2.4.1: *Pricing RMSE (absolute pricing errors) and training times without Dupire penalization.*

Figure 2.4.3 compares the percentage errors in implied volatilities using the sparse network with hard constraints and the dense network with soft constraints approaches, corresponding to the columns 1 and 3 of Table 2.4.1. Relative errors with hard constraints exceed 10% on most the training grid oppositely to dense network with soft constraints. This confirms that the error levels of the hard constraints approach are too high to imagine a practical use of this approach: the corresponding model would be immediately arbitrable in relation to the market. Those of the soft constraint approach are much more acceptable, with high errors confined to short maturities or far from the money, i.e. in the region where prices provide little information on volatility.

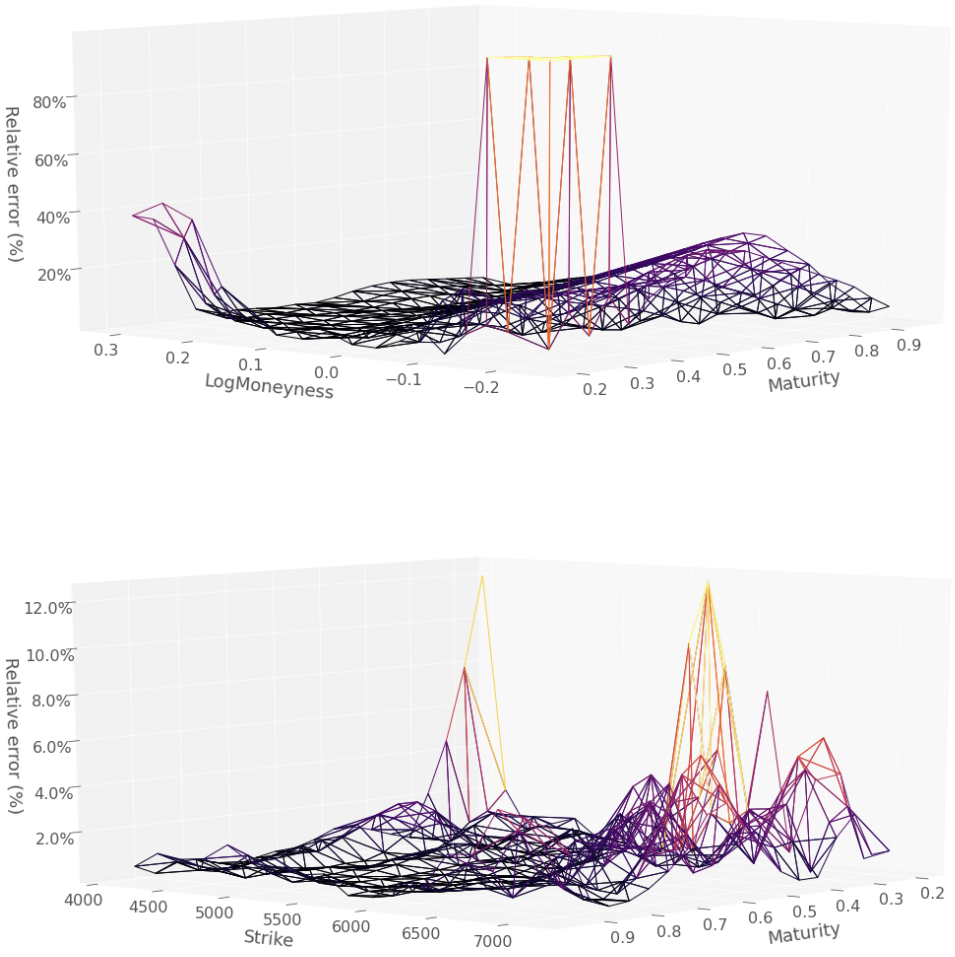


Figure 2.4.3: *Percentage relative error in the implied volatilities using (top) hard constraints (bottom) dense networks with soft constraints.*

Table 2.4.2 shows the fraction of points in the neural network price surface which violate the static arbitrage conditions. The table compares the same four methods listed in Table 2.4.3 applied to training and testing sets. We recall that, in theory, only the sparse network with hard constraints guarantees zero arbitrages. However, we observe that the inclusion of soft constraints reduces the number of arbitrage constraints on the training set when compared with no constraints. The trend is less pronounced for the test set. But in the absence of hard constraints, the effect of adding soft constraints is always preferable than excluding them entirely.

	Sparse network		Dense network	
	Hard constraints	Soft constraints	Soft constraints	No constraints
Training dataset	0	1/254	0	63/254
Testing dataset	0	2/360	0	44/360

Table 2.4.2: *The fraction of static arbitrage violations without Dupire penalization.*

2.4.3 Numerical Results With Dupire Penalization

We now introduce half-variance bounds into the penalization to improve the overall fit in prices and stabilize the local volatility surface. Table 2.4.3 shows the RMSEs in absolute pricing resulting from repeating the same set of experiments reported in Table 2.4.1, but with the half-variance bounds included in the penalization. For the sparse network with hard constraints, we set $\lambda = [0, 0, 10]$ and choose $\underline{a} = 0.05^2/2$ and $\overline{a} = 0.4^2/2$. For the sparse and dense networks with soft constraints, we set $\lambda = [1.0 \times 10^5, 1.0 \times 10^3, 10]$. Compared to Table 2.4.1, we observe improvement in the test error for the hard and soft constraints approaches when including the additional local volatility penalty term. Table 2.4.4 is the analog of Table 2.4.2, with similar conclusions. Note that, here as above, the arbitrage opportunities that arise are not only very few (except in the unconstrained case), but also very far from the money and, in fact, mainly regard the learning of the payoff function, corresponding to the horizon $T = 0$. See for instance Figure 2.4.4 for the location of the violations that arise in the unconstrained case with Dupire penalization. Hence such apparent 'arbitrage opportunities' cannot necessarily be monetised once liquidity is accounted for.

Figure 2.4.6 is the analog of Figure 2.4.1, with test (i.e. Tikhonov trinomial tree) prices in blue replaced by the prices predicted by the dense network with soft constraints and Dupire penalization. The (blue) prices predicted by the neural network in Figure 2.4.6, and the corresponding implied volatilities in Figure 2.4.7, do not exhibit any visible inter-extrapolation pathologies, they are in fact visually indistinguishable from the respective (blue) testing prices and implied volatilities of Figures 2.4.1 and 2.4.2.

	Sparse network		Dense network	
	Hard constraints	Soft constraints	Soft constraints	No constraints
Training dataset	28.04	3.44	2.48	3.48
Testing dataset	27.07	3.33	3.36	4.31
Indicative training times	400s	600s	300s	250s

Table 2.4.3: *Price RMSE (absolute pricing errors) and training times with Dupire penalization.*

	Sparse network		Dense network	
	Hard constraints	Soft constraints	Soft constraints	No constraints
Training dataset	0	0	0	30/254
Testing dataset	0	2/360	0	5/360

Table 2.4.4: *The fraction of static arbitrage violations with Dupire penalization.*

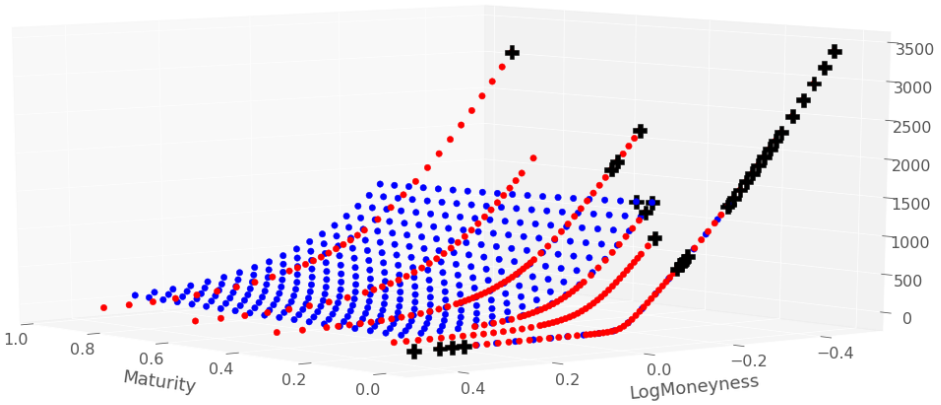


Figure 2.4.4: *Location of the violations, denoted by black crosses, corresponding to the right column in Table 2.4.4.*

For completeness, we additionally provide further diagnostic results. Figure 2.4.5 shows the convergence of the loss function against the number of epochs using either hard constraints or soft constraints. The spikes trigger decays of the learning rates so that the training procedure can converge toward a local minimum of the loss criterion (cf. Section 2.4.1). We observe that the loss function converges to a much smaller value using a dense network with soft constraints and that either approach converge in at most 2000 epochs.

Table 2.4.5 provides some further insight into the effect of architectural param-

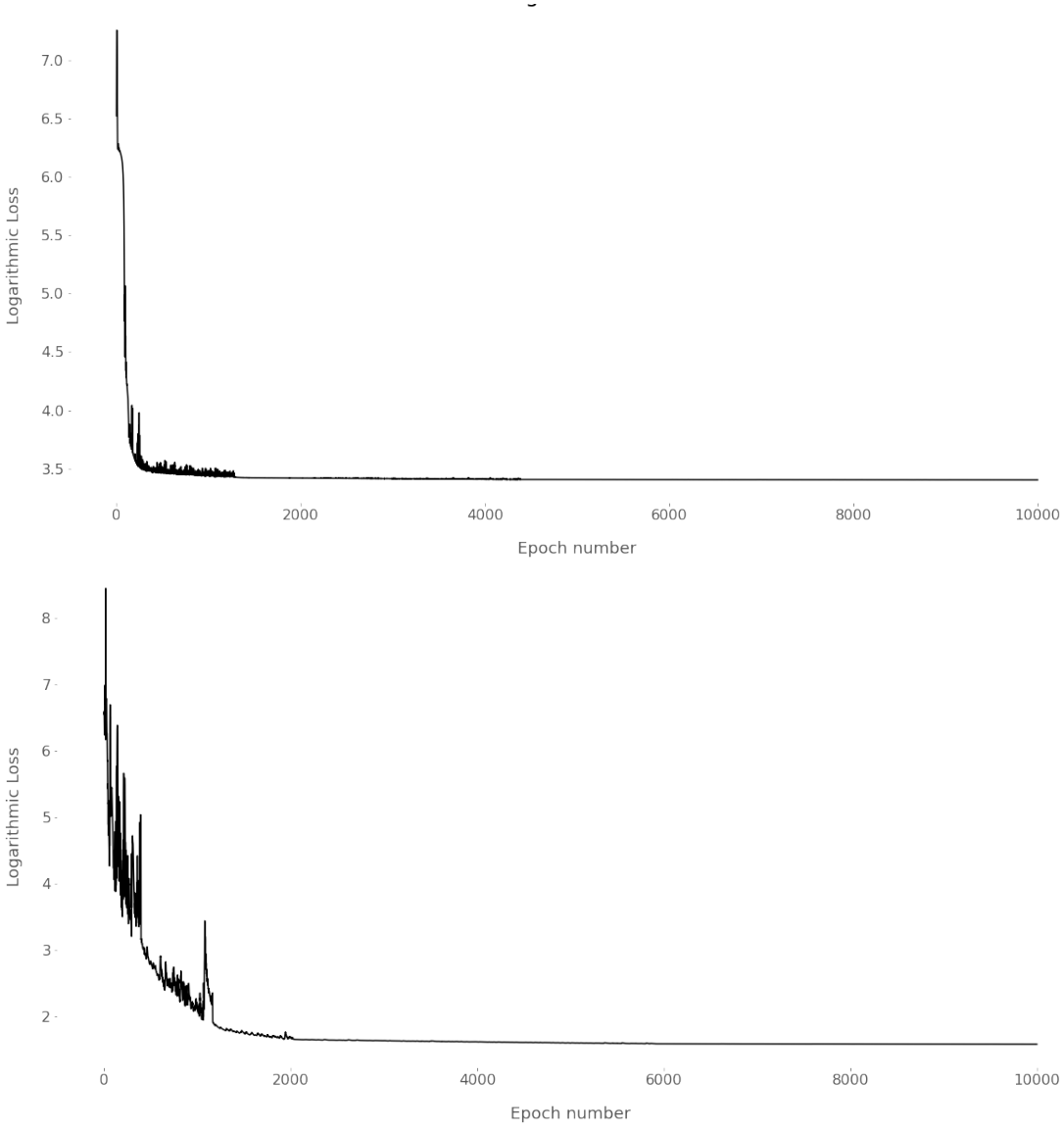


Figure 2.4.5: *Logarithmic RMSE through epochs (top) hard constraints (bottom) dense networks with soft constraints.*

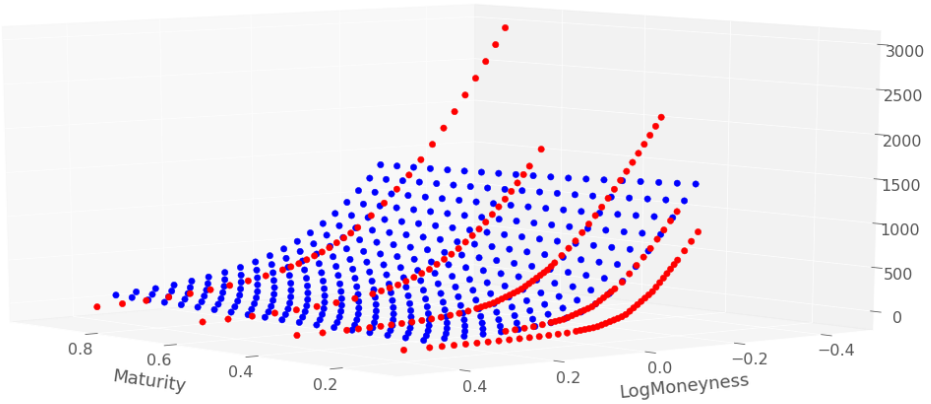


Figure 2.4.6: *Put prices from training grid (red points) and NN predicted prices at testing grid nodes (blue points), DAX 8 Aug 2001.*

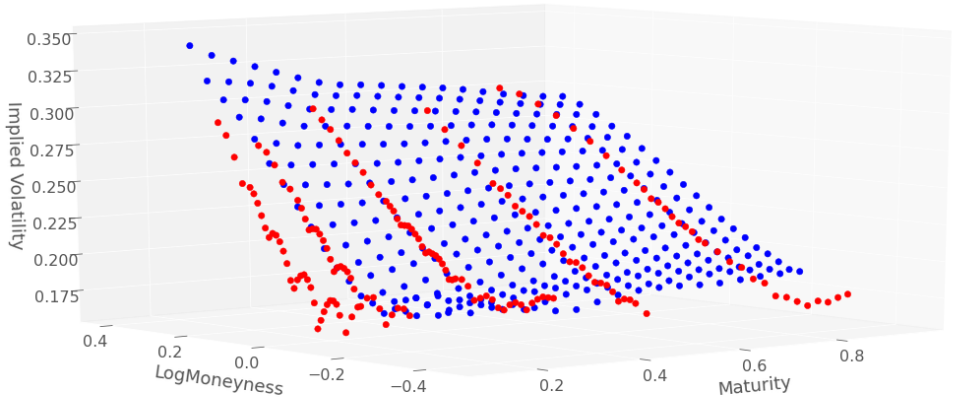


Figure 2.4.7: Same as Figure 2.4.6 in implied volatility scale.

eters, although it is not intended to be an exhaustive study. Here, only the number of units in the hidden layers is varied, while keeping all other parameters except the learning rate fixed, to study the effect on error in the price and implied volatility surfaces. The price RMSE for the testing set primarily provides justification for the choice of 200 hidden units per layer: the RMSE is 3.55. We further observe the effect of reduced pricing error on the implied volatility surface: 0.0036 is the lowest RMSE of the implied volatility test surface across all parameter values.

# Hidden Units	Surface	RMSE	
		Training	Testing
50	Price	3.01	3.60
	Impl. Vol.	0.0173	0.0046
100	Price	3.14	3.66
	Impl. Vol.	0.0304	0.0049
200	Price	2.73	3.55
	Impl. Vol.	0.0181	0.0036
300	Price	2.84	3.88
	Impl. Vol.	0.0180	0.0050
400	Price	2.88	3.56
	Impl. Vol.	0.0660	0.0798

Table 2.4.5: *Sensitivity of the errors to the number of hidden units. Note that these results are generated using the dense network with soft constraints and Dupire penalization.*

Table 2.4.6 shows the pricing RMSEs resulting from the application of different stochastic gradient descent algorithms under the soft constraints approach with dense network and Dupire penalization. ADAM (our choice everywhere else in the chapter, cf. the next-to-last column in Table 2.4.3) and RMSProp (root mean square propagation, another well known SGD procedure) exhibit a comparable performance. A Nesterov accelerated gradient procedure, with momentum parameter set to 0.9 as standard, obtains much less favorable results. As opposed to ADAM and RMSProp, Nesterov accelerated momentum does not reduce the learning rate during the optimization.

2.4.4 Robustness

In this concluding section of the DAX experiments, we assess the robustness of the different approaches in terms of, first, the numerical stability of the local volatility function recalibrated across successive calendar days and, second, of a Monte Carlo backtesting repricing error.

	Train RMSE	Test RMSE
ADAM	2.48	3.36
Nesterov accelerated gradient	5.67	6.92
RMSProp	2.76	3.66

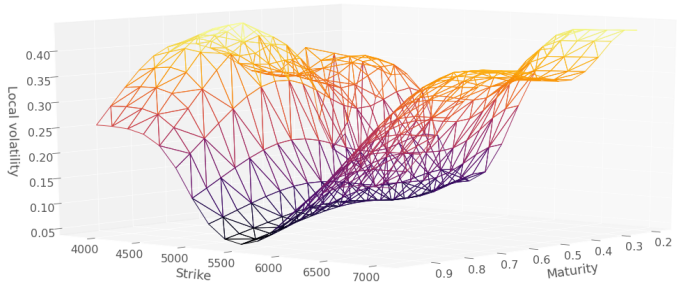
Table 2.4.6: Pricing RMSEs corresponding to different stochastic gradient descents (soft constraints approach with dense network and Dupire penalization).

Numerical Stability Through Recalibration

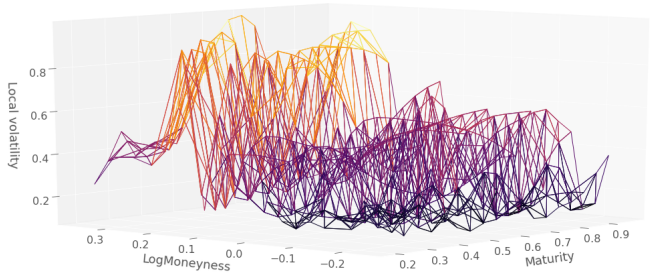
Figures 2.4.8, 2.4.9 and 2.4.10 show the comparison of the local volatility surfaces obtained using hard constraints (sparse network) without Dupire penalization, dense network and soft constraints without and with Dupire penalization, as well as the Tikhonov regularization approach of (Crépey 2002), on price quotes listed on August 7^{th}, 8^{th}, and 9^{th}, 2001, respectively. The soft constraint approach without Dupire penalization is both irregular (exhibiting outliers on a given day) and unstable (from day to day). In contrast, the soft constraint approach with Dupire penalization yields a more regular (at least, less spiky) local volatility surface, both at fixed calendar time and in terms of stability across calendar time. From this point of view the results are then qualitatively comparable to those obtained by Tikhonov regularization (which is however quicker, taking of the order of 30s to run).

Monte Carlo Backtesting Repricing Error

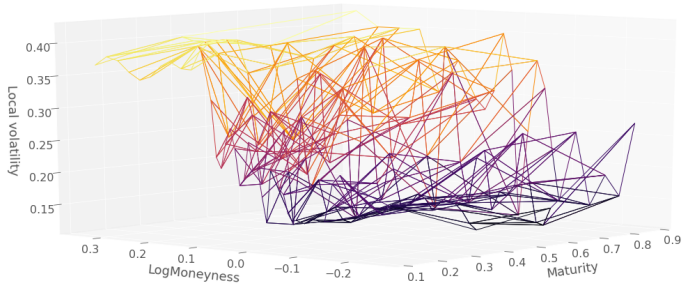
Finally, we evaluate the performance of the models in a backtesting Monte Carlo exercise. Namely, the options in each testing grid are repriced by Monte Carlo with 10^5 paths of 100 time steps in the model 2.7 using differently calibrated local volatility functions $\sigma(\cdot, \cdot)$ in (2.7), for each of the 7th, 8th, and 9th August dataset. Table 2.4.7 shows the corresponding Monte Carlo backtesting repricing errors, using the option market prices from the training grids as reference values in the corresponding RMSEs. The neural network approaches provide a full surface of prices and local volatilities, as opposed to values at the calibration trinomial tree nodes only in the case of Tikhonov, for which the Monte Carlo backtesting exercise thus requires an additional layer of local volatility inter-extrapolation, here achieved by a nearest neighbors algorithm. We see from the table that both the benchmark Tikhonov method and the dense network soft constraints approach with Dupire penalization yield very reasonable and acceptable repricing errors (with still a certain advantage to the Tikhonov method), unlike the hard constraints approaches. Moreover, the Dupire penalization is essential for extracting a decent local volatility function: The dense network with soft constraint but without this penalization



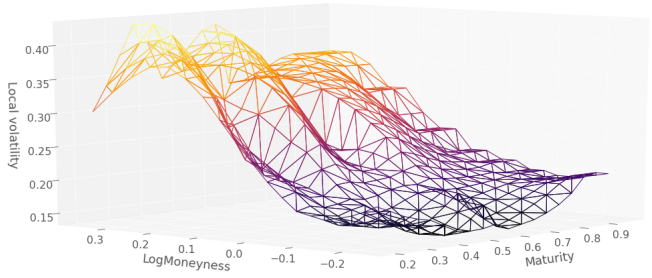
(a) Hard Constraints



(b) Soft constraints (w/o local vol. constraints)

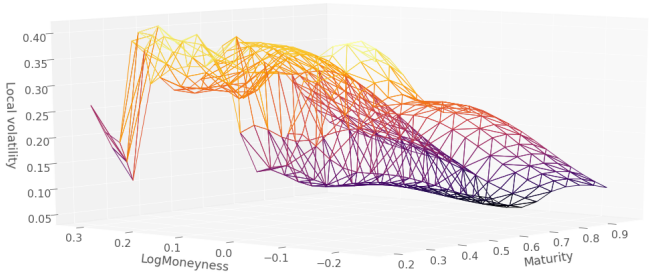


(c) Soft constraints (with local vol. constraints)

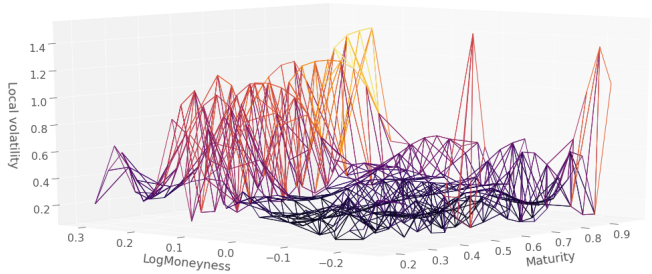


(d) Tikhonov local volatility

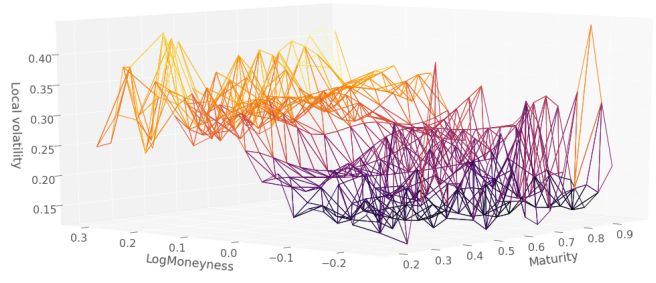
Figure 2.4.8: *Local volatility for 07/08/2001.*



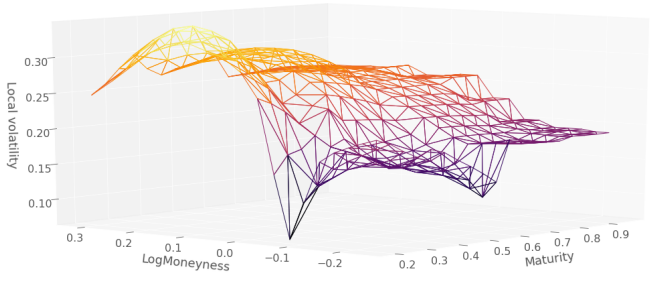
(a) Hard Constraints



(b) Soft constraints (w/o local vol. constraints)

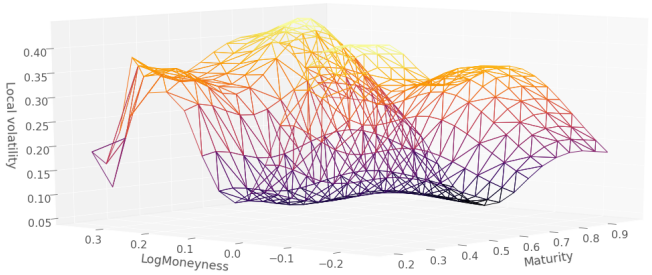


(c) Soft constraints (with local vol. constraints)

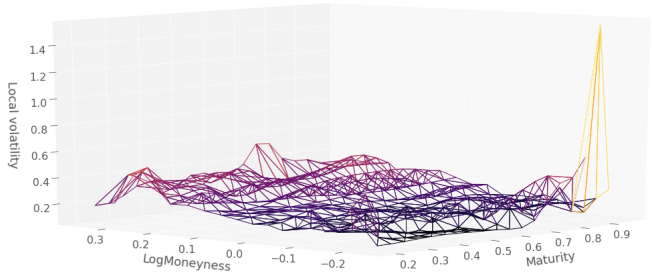


(d) Tikhonov local volatility

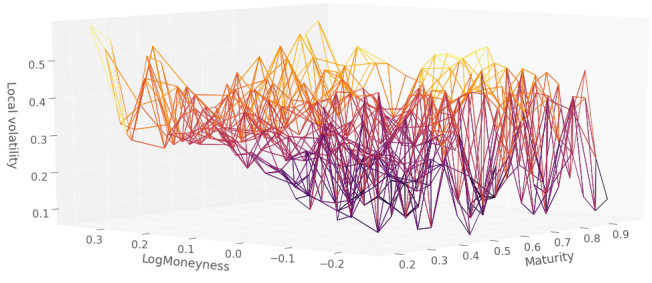
Figure 2.4.9: *Local volatility for 08/08/2001.*



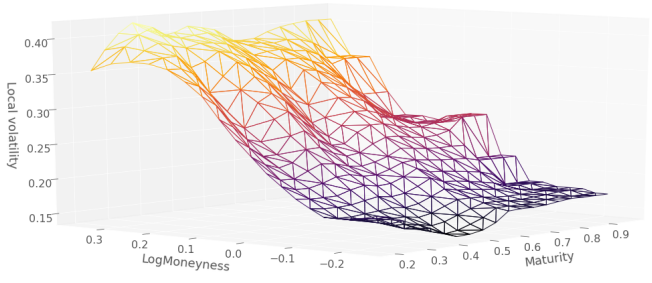
(a) Hard Constraints



(b) Soft constraints (w/o local vol. constraints)



(c) Soft constraints (with local vol. constraints)



(d) Tikhonov local volatility

Figure 2.4.10: *Local volatility for 09/08/2001.*

yields very poor Monte Carlo repricing RMSEs.

$\sigma(\cdot,\cdot)$	Tikhonov Monte Carlo	Dense network with soft constraints and Dup. penal.	Dense network with soft constraints	Hard constraint with Dup. penal.	Hard constraint w/o Dup. Pen.
07/08/2001	5.42	10.18	68.48	48.57	50.44
08/08/2001	5.55	7.44	50.82	56.63	56.98
09/08/2001	4.60	8.18	59.39	66.23	65.50

Table 2.4.7: Monte Carlo backtesting repricing RMSEs on training grid against market prices.

The residual gap between the Monte Carlo RMSEs of the (even best) neural network local volatility and of the Tikhonov local volatility can seem disappointing. However we should keep in mind that the neural network can evaluate quickly a local volatility on any node outside the training grid, whereas Tikhonov then requires a further layer of interpolation (or a new calibration). Furthermore, any vanilla option price can be accurately and quickly obtained by neural prediction (better than by Monte Carlo repricing as above). Table 2.4.8 shows training set RMSEs of thus predicted prices against markets prices equivalent to (in fact, slightly better than) RMSEs of Tikhonov trinomial tree prices against the same markets prices. These are of course only in-sample errors, but the additional findings of Table 2.4.7 suggest that these good results are not just overfitting.

$\sigma(\cdot,\cdot)$	Tikhonov trin. tree	NN pred. (dense network with soft constraints and Dup. penal).
07/08/2001	2.42	2.66
08/08/2001	2.67	2.48
09/08/2001	2.45	2.34

Table 2.4.8: Training set RMSEs of Tikhonov trinomial tree vs. NN predicted prices against market prices.

2.5 Gaussian process regression for learning arbitrage-free price surfaces

In this section, we consider as a first benchmark a zero-mean Gaussian process prior on the mapping $P = P(x)_{x\in\Omega}$ with correlation function c given, for any

$x = (T, k), x' = (T', k') \in \Omega$, by

$$c(x, x') = \sigma^2 \gamma_T(T - T', \theta_T)\gamma_k(k - k', \theta_k). \tag{2.8}$$

The introduction of Gaussian processes as an arbitrage-free surrogate model is a recent proposal. Tegnér & Roberts (Tegnér and Roberts 2019, see their Eq. (10)) first attempt the use of GPs for local volatility modeling by placing a Gaussian prior directly on the local volatility surface. Such an approach leads to a nonlinear least squares training loss function, which is not obviously amenable to gradient descent (stochastic or not), so the authors resort to a MCMC optimization. Maatouk & Bay (Maatouk and Bay 2017) introduce finite dimensional approximation of Gaussian processes (GP) for which shape constraints are straightforward to impose and verify. Cousin et al. (Cousin, Maatouk, and Rullière 2016b) apply this technique to ensure arbitrage-free and error-controlled yield-curve and CDS curve interpolation and below we present their most recent approach detailed in (Chataigner, Cousin, Crépey, Dixon, and Gueye 2021).

Here $(\theta_T, \theta_k) = \theta$ and σ^2 correspond to length scale and variance hyper-parameters of the kernel function c, whereas the functions γ_T and γ_k are kernel correlation functions.

Without consideration of the conditions (2.2), (unconstrained) prediction and uncertainty quantification are made using the conditional distribution $P \mid P(\mathbf{x}) + \boldsymbol{\varepsilon} = \boldsymbol{y}$, where $\boldsymbol{y} = [y_1, \ldots, y_n]^\top$ are n noisy observations of the function P at input points $\mathbf{x} = [x_1, \ldots, x_n]^\top$, corresponding to observed maturities and strikes $x_i = (T_i, k_i)$; the additive noise term $\boldsymbol{\varepsilon} = [\varepsilon_1, \ldots, \varepsilon_n]^\top$ is assumed to be a zero-mean Gaussian vector, independent from $P(\mathbf{x})$, and with an homoscedastic covariance matrix given as $\varsigma^2 I_n$, where I_n is the identity matrix of dimension n. Note that bid and ask prices are considered here as (noisy) replications at the same input location.

2.5.1 Imposing the no-arbitrage conditions

To deal with the constraints (2.2), we adopt the solution of Cousin et al. (Cousin, Maatouk, and Rullière 2016b) that consists in constructing a finite dimensional approximation P^h of the Gaussian prior P for which these constraints can be imposed in the entire domain Ω with a finite number of checks. One then recovers the (non Gaussian) constrained posterior distribution by sampling a truncated Gaussian process.

Remark 1. *Switching to a finite dimensional approximation can also be viewed as a form of regularization, which is also required to deal with the ill-posedness of the (numerical differentiation) Dupire formula.*

We first consider a discretized version of the (rescaled) input space $\Omega = [0, 1]^2$ as a regular grid $(\imath h)_\imath$, where $\imath = (i, j)$, for a suitable mesh size h and indices i, j

ranging from 0 to $1/h$ (taken in $\mathbb{N}^\star$). For each knot $\imath = (i, j)$, we introduce the hat basis functions $\phi_\imath$ with support $[(i-1)h, (i+1)h] \times [(j-1)h, (j+1)h]$ given, for $x = (T, k)$, by

$$\phi_\imath(x) = \max(1 - \frac{|T - ih|}{h}, 0) \max(1 - \frac{|k - jh|}{h}, 0).$$

We take $V = H^1(\Omega) = \{u \in L_2(\Omega) : D^\alpha u \in L_2(\Omega), |\alpha| \leq 1\}$, where $D^\alpha u$ is a weak derivative of order $|\alpha|$, as the space of (the realizations of) P. Let $V^h \subset V$ denote the finite dimensional linear subspace spanned by the M linearly independent basis functions $\phi_\imath$. The (random) surface P in V is projected onto V^h as

$$P^h(x) = \sum_\imath P(\imath h)\phi_\imath(x), \ \forall x \in \Omega. \tag{2.9}$$

If we denote $L_\imath = P(\imath h)$, then $\boldsymbol{L} = (L_\imath)_\imath$ is a zero-mean Gaussian column vector (indexed by $\imath$) with $M \times M$ covariance matrix Γ^h such that $\Gamma^h_{\imath,\jmath} = c(\imath h, \jmath h)$, for any two grid nodes $\imath$ and $\jmath$. Let $\boldsymbol{\phi}(x)$ denote the vector of size M given by $\boldsymbol{\phi}(x) = (\phi_\imath(x))_\imath$. The equality (2.9) can be rewritten as $P^h(x) = \boldsymbol{\phi}(x) \cdot \boldsymbol{L}$. Denoting by $P^h(\mathbf{x}) = [P^h(x_1), \ldots, P^h(x_n)]^\top$ and by $\boldsymbol{\Phi}(\mathbf{x})$ the $n \times M$ matrix of basis functions where each row ℓ corresponds to the vector $\boldsymbol{\phi}(x_\ell)$, one has $P^h(\mathbf{x}) = \boldsymbol{\Phi}(\mathbf{x}) \cdot \boldsymbol{L}$. By application of the results of (Maatouk and Bay 2017):

Proposition 2. (i) *The finite dimensional process P^h converges uniformly to P on Ω as $h \to 0$, almost surely,*
(ii) *$P^h(T, k)$ is a nondecreasing function of T if and only if $L_{i+1,j} \geq L_{i,j}, \forall (i, j)$,*
(iii) *$P^h(T, k)$ is a convex function of k if and only if $L_{i,j+2} - L_{i,j+1} \geq L_{i,j+1} - L_{i,j}, \forall (i, j)$.* ∎

In view of (i), denoting by $\mathcal{I}$ the set of 2d continuous positive functions which are nondecreasing in T and convex in k, we choose as constrained GP metamodel for the put price surface the law of P^h conditional on

$$\begin{cases} P^h(\mathbf{x}) + \boldsymbol{\varepsilon} = \boldsymbol{y} \\ P^h \in \mathcal{I}. \end{cases}$$

In view of (ii)-(iii), $P^h \in \mathcal{I}$ if and only if $\boldsymbol{L} \in \mathcal{I}^h$, where $\mathcal{I}^h$ corresponds to the set of ($\imath$ indexed) vectors $\boldsymbol{\rho} = (\rho_\imath)_\imath$ such that $\rho_{i+1,j} \geq \rho_{i,j}$ and $\rho_{i,j+2} - \rho_{i,j+1} \geq \rho_{i,j+1} - \rho_{i,j}$ $\forall (i, j)$. Hence, our GP metamodel for the put price surface can be reformulated as the law of $\boldsymbol{L}$ conditional on

$$\begin{cases} \boldsymbol{\Phi}(\mathbf{x}) \cdot \boldsymbol{L} + \boldsymbol{\varepsilon} = \boldsymbol{y} \\ \boldsymbol{L} \in \mathcal{I}^h. \end{cases} \tag{2.10}$$

2.5.2 Hyper-parameter learning

Hyper-parameters consist in the length scales θ and the variance parameter σ^2 in (2.8), as well as the noise variance ς. Up to a constant, the so called marginal log likelihood of $\boldsymbol{L}$ at $\lambda = [\theta, \sigma, \varsigma]^\top$ can be expressed as (see e.g. (Murphy 2012a, Section 15.2.4, p. 523)):

$$\mathcal{L}(\lambda) = -\tfrac{1}{2}\boldsymbol{y}^\top \left(\boldsymbol{\Phi}(\mathbf{x})\Gamma^h\boldsymbol{\Phi}(\mathbf{x})^\top + \varsigma^2 I_n\right)^{-1}\boldsymbol{y} - \tfrac{1}{2}\log\Big(\det\left(\boldsymbol{\Phi}(\mathbf{x})\Gamma^h\boldsymbol{\Phi}(\mathbf{x})^\top + \varsigma^2 I_n\right)\Big).$$

We maximize $\mathcal{L}$ for learning the hyper-parameters λ (MLE estimation).

Remark 3. *The above expression does not take into account the inequality constraints in the estimation. However, Bachoc et al. (Bachoc, Lagnoux, López-Lopera, et al. 2019, see e.g. their Eq. (2)) argue (and we observed empirically) that, unless the sample size is very small, conditioning by the constraints significantly increases the computational burden with negligible impact on the MLE.*

2.5.3 The most probable response surface and measurement noises

We compute the joint MAP $(\hat{\boldsymbol{\rho}}, \hat{\boldsymbol{e}})$ of the truncated Gaussian vector $\boldsymbol{L}$ and of the Gaussian noise vector $\boldsymbol{\varepsilon}$,

$$(\hat{\boldsymbol{\rho}}, \hat{\boldsymbol{e}}) = \underset{(\boldsymbol{\rho},\boldsymbol{e})}{\arg\max}\ \mathrm{Prob}\left(\boldsymbol{L} \in [\boldsymbol{\rho}, \boldsymbol{\rho} + \mathrm{d}\boldsymbol{\rho}], \boldsymbol{\varepsilon} \in [\boldsymbol{e}, \boldsymbol{e} + \mathrm{d}\boldsymbol{e}] \mid \boldsymbol{\Phi}(\mathbf{x}) \cdot \boldsymbol{L} + \boldsymbol{\varepsilon} = \boldsymbol{y},\ \boldsymbol{L} \in \mathcal{I}^{\mathrm{h}}\right)$$

(for the probability measure Prob underlying the GP model). As $(\boldsymbol{L}, \boldsymbol{\varepsilon})$ is Gaussian centered with block-diagonal covariance matrix with blocks Γ^h and $\varsigma^2 I_n$, this implies that the MAP $(\hat{\boldsymbol{\rho}}, \hat{\boldsymbol{e}})$ is a solution to the following quadratic problem :

$$\underset{\boldsymbol{\Phi}(\mathbf{x})\cdot\boldsymbol{\rho}+\boldsymbol{e}=\boldsymbol{y},\ \boldsymbol{\rho}\in\mathcal{I}^h}{\arg\min} \left(\boldsymbol{\rho}^\top(\Gamma^h)^{-1}\boldsymbol{\rho} + \boldsymbol{e}^\top(\varsigma^2 I_n)^{-1}\boldsymbol{e}\right). \tag{2.11}$$

We define the most probable measurement noise to be $\hat{\boldsymbol{e}}$ and the most probable response surface $\hat{p}^h(\mathbf{x}) = \boldsymbol{\Phi}(\mathbf{x})\cdot\hat{\boldsymbol{\rho}}$. Distance to the data can be an effect of arbitrage opportunities within the data and/or misspecification / lack of expressiveness of the kernel.

2.5.4 Sampling finite dimensional Gaussian processes under shape constraints

The conditional distribution of $\boldsymbol{L} \mid \boldsymbol{\Phi}(\mathbf{x}) \cdot \boldsymbol{L} + \boldsymbol{\varepsilon} = \boldsymbol{y}$ is multivariate Gaussian with mean $\boldsymbol{\eta}_{\mathbf{y}}(\mathbf{x})$ and covariance matrix $\mathbf{C}_{\mathbf{y}}(\mathbf{x})$ such that

$$\boldsymbol{\eta}_{\mathbf{y}}(\mathbf{x}) = \Gamma^h\boldsymbol{\Phi}(\mathbf{x})^\top(\boldsymbol{\Phi}(\mathbf{x})\Gamma^h\boldsymbol{\Phi}(\mathbf{x})^\top + \varsigma^2 I_n)^{-1}\boldsymbol{y} \tag{2.12}$$

$$\mathbf{C}_{\mathbf{y}}(\mathbf{x}) = \Gamma^h\boldsymbol{\Phi}(\mathbf{x})^\top(\boldsymbol{\Phi}(\mathbf{x})\Gamma^h\boldsymbol{\Phi}(\mathbf{x})^\top + \varsigma^2 I_n)^{-1}\boldsymbol{\Phi}(\mathbf{x})\Gamma^h. \tag{2.13}$$

In view of (2.10), we thus face the problem of sampling from this truncated multivariate Gaussian distribution, which we do by Hamiltonian Monte Carlo, using the MAP $\hat{\boldsymbol{L}}$ of $\boldsymbol{L}$ as the initial vector (which must verify the constraints) in the algorithm.

2.5.5 Local volatility

Due to the shape constraints and to the ensuing finite-dimensional approximation with basis functions of class $\mathcal{C}^0$ (for the sake of Proposition 2), P^h is not differentiable. Hence, exploiting GP derivatives analytics, as done for the mean in (Crépey and Dixon 2020, cf. Eq. (10)) and also for the covariance in (Ludkovski and Saporito 2020), is not possible for deriving the corresponding local volatility surface here. Computation of derivatives involved in the Dupire formula is implemented by finite differences with respect to a coarser grid (than the grid of basis functions). Another related solution would be to formulate a weak form of the Dupire equation and construct a local volatility surface approximation using a finite element method.

See Algorithm 2.1 for the main steps of the GP approach.

Algorithm 3 The GP algorithm for local volatility surface approximation.

Data: Put price training set $P_\star$

Result: M realizations of the local volatility surface $\{\mathrm{dup}_i^h\}_{i=1}^M$

$\hat{\lambda} \leftarrow$ Maximize the marginal log-likelihood of the put price surface p^h w.r.t. λ `// Hyperparameter fitting`

$(\hat{\boldsymbol{\rho}}, \hat{\boldsymbol{e}}) \leftarrow$ Minimize quadratic problem (2.11) based on $\hat{\lambda}$ `// Joint MAP estimate`

$\hat{\boldsymbol{\rho}} \rightarrow$ Initialize a Hamiltonian MC sampler

$P_1^h, \ldots, P_M^h \leftarrow$ Hamiltonian MC Sampler `// Sampling price surfaces`

$\mathrm{dup}_i^h \leftarrow$ Finite difference approximation using each $P_i^h,\ i := 1 \rightarrow M$

2.6 Arbitrage-free SVI

2.6.1 SVI parameterizations

We also benchmark the machine learning results with the industry standard provided by the arbitrage free stochastic volatility inspired (SVI) model of (Gatheral and Jacquier 2014). Under the "natural parameterization" SVI $= (\Delta, \mathbb{P}, \rho, \omega, \zeta)$, the implied total variance is given, for any fixed T, by

$$\Theta^{\mathrm{SVI}}(\kappa) = \Delta + \frac{\omega}{2}\left(1 + \rho(\kappa - \mathbb{P})\zeta + \sqrt{(\zeta(\kappa - \mathbb{P}) + \rho)^2 + (1 - \rho^2)}\right). \qquad (2.14)$$

In order to explain the arbitrage-free SSVI methodology, we need to introduce the companion parameterizations. The raw SVI parameterization was the first SVI model introduced at Merryl Lynch in 1999 and then reported in the litterature (see (Gatheral 2004)). Under this raw parameterization, the implied total variance has the advantage , for a fixed maturity , to be a linear function of the log-moneyness for large strike which is consistent with the Roger Lee's moment formula.

This raw parameterization (a, b, L, m, σ) is in bijection with the natural parameterization :

$$\begin{cases} a = & \Delta + \frac{\omega}{2}(1 - L^2) \\ L = & \rho \\ b = & \frac{\omega\zeta}{2} \\ m = & \mathbb{P} - \frac{L}{\zeta} \\ \sigma = & \frac{\sqrt{1-L^2}}{\zeta} \end{cases} \tag{2.15}$$

The Jump-Wings SVI prametrization $(v, p, \psi, c, \hat{v})$ is more intuitive in so far as its parameters are closely related to the geometry of the slice. For instance v yields the at-the-money implied total variance, ψ gives the ATM skew, p and c the slope for respectively the left put and the right call wing and $\hat{v}$ the minimum implied variance.

The Jump-Wings SVI parameterization is also in bijection with the SVI raw parameterization $\forall t \geq 0$:

$$\begin{cases} b = & \frac{\sqrt{w}}{2(c+p)} \\ L = & 1 - \frac{p\sqrt{w}}{b} \\ a = & L - \frac{2\psi\sqrt{w}}{b} \\ m = & \frac{(v-\hat{v})t}{b\left(-L+sign(\alpha)\sqrt{1+\alpha^2}-\alpha\sqrt{1-\alpha^2}\right)} \\ \sigma = & \alpha m \end{cases} \tag{2.16}$$

with $w = vt$, $\alpha = sign(\beta)\sqrt{\frac{1}{\beta^2} - 1}$ and $\beta = L - \frac{2\psi\sqrt{w}}{b}$.

2.6.2 No-arbitrage conditions on SVI parameters

Arbitrage Free surface SVI model requires monotonicity of at-the-money implied total variance. In our data some slices are unbalanced and does not contain at-the-money logmoneyness in their domain. This leads to the estimation of ATM implied total variance breaking that monotony. To circumvent that problem, we estimate ATM implied total variance only with slices containing zero forward log-moneyness. Then we interpolate linearly that implied total variance and we ensure

monotonicity by taking for each t the maximum of previous ATM total variance $\max_{s\leq t}\Theta(s,0)$:

$$\Theta(t,0) = \begin{cases} \max_{s\leq t}\Theta(s,0) & \text{if } 0 \in [\min_{\{\kappa \text{ s.t. } \chi_i(t,\kappa)\in\Omega\}}\kappa, \max_{\{\kappa \text{ s.t. } \chi_i(t,\kappa)\in\Omega\}}\kappa] \\ \frac{t-t^-}{t^+-t^-}\min_{\left\{s\geq t^+ \text{ , s.t. } \Theta(s,0)\leq \max_{s\leq t^-}\Theta(s,0)\right\}}\Theta(s,0) + \frac{t^+-t}{t^+-t^-}\max_{s\leq t^-}\Theta(s,0) & \text{otherwise} \end{cases} \tag{2.17}$$

The first step for an arbitrage-free SVI slice is to get an arbitrage-free initial guess. This is done for all slice with a single calibration of Surface SVI model (or SSVI). SSVI is a particular case of natural SVI parameterization namely $(0, 0, \rho, \Theta(t,0), \phi(\Theta(t,0)))$ where ϕ is the so-called "power law" function :

$$\phi(\Theta) = \frac{\eta}{\Theta^\gamma(1+\Theta)^{1-\gamma}}.$$

When $\gamma = 0.5$, (Gatheral and Jacquier 2014, Remark 4.4) shows that no-arbitrage conditions are respected under the constraint $\eta(1+|L|)-2 \geq 0$ (assuming that the ATM implied total variance is also nondecreasing).

SSVI model is thus calibrated on training dataset through the following optimization problem :

$$SSVI \leftarrow \underset{\substack{(\eta,L)\\ \eta(1+|L|)-2\geq 0}}{\arg\min} \sqrt{\frac{1}{n}\sum_i \left(\left(\Sigma_{SVI_t^{nat}}(\chi_i,\Theta(t,0),L,\eta) - \Sigma_*(\chi_i)\right)\right)^2}, \tag{2.18}$$

The subsequent initial guess $SSVI$ is then plugged in an optimization routine (2.19) specific to each slice. The calendar condition is maintained between two consecutive slices with a crossedness penalty. Denoting by SVI_t the arbitrage-free SVI parameters for a slice of implied volatility at time t, the crossedness penalty is defined as : $\mathcal{C}(t) = \max_{\kappa_j}\left(\Theta^{SVI}(t^-,\kappa_j) - \Theta^{SVI}(t,\kappa_j)\right)^+$ where $(\kappa_j)_{\{j\in\{1,..,n\}\}}$ is an arbitrary forward log-moneyness discretization.

For each maturity t in the training set we solve the following optimization problem :

$$SVI_t \leftarrow \underset{(v,p,\psi)}{\arg\min} \sqrt{\frac{1}{n}\sum_i \left(\left(\Sigma_{SVI_t^{JW}}(\chi_i) - \Sigma_*(\chi_i)\right)\right)^2} + \lambda\mathcal{C}(t), \tag{2.19}$$

where $\lambda \in \mathbb{R}_+$ and SVI_t^{JW} employs the SVI-Jump-Wings parameterization at time t $(v, p, \psi, \psi + 2p, \frac{4v\psi(\psi+2p)}{(\psi+\psi+2p)^2})$.

2.6.3 Slice parameter interpolation

When we are looking for an implied total variance with a maturity outside the training set, we need to interpolate the SVI parameters of the two nearest slices. Let denote t^- and t^+ the closest maturities in the training set surrounding an arbitrary date t. Let us assume that parameters SVI_{t^+} and SVI_{t^-} are arbitrage-free. Then the following interpolation guarantee an arbitrage-free SVI slice :

$$SVI_t = \alpha_t SVI_{t^+} + (1 - \alpha_t) SVI_{t^-}$$

with $\alpha_t = \frac{\Theta(t,0)-\Theta(t^-,0)}{\Theta(t^+,0)-\Theta(t^-,0)}$ and $\Theta(.,0)$ the ATM implied total variance curve.

When t is greater than the largest maturity in the training set, $\Theta(t,.)$ is extrapolated as follow :

$$\Theta(t,\kappa) = \Theta(t^-,\kappa) + \hat{\Theta}(t,0) - \Theta(t^-,0), \forall \kappa$$

with $\hat{\Theta}(t,0)$ is the ATM implied total variance linearly extrapolated from ATM implied total variance of the training set.

If t is smaller than the smallest training maturity, implied volatility is calibrated on a combination of payoff $P_0(k)$ and price P_{t^+}. In our case $\Theta(t,k)$ is obtained with a bissection algorithm on interpolated price $P_t(k)$:

$$P_t(k) = \alpha_t \frac{P_0(k)}{K_0} + (1 - \alpha_t) \frac{P_{t^+}}{K_{t^+}}$$

with $\alpha_t = \frac{\Theta(t^+,0)-\Theta(t,0)}{\Theta(t,0)}$ and $K_t = e^{-k} F_t$ where F_t is the forward price at time t.

Once the SVI slices are fitted, the corresponding local volatility is extracted by finite difference approximation of (2.3). As, in practice, no arbitrage constraints are implemented for SSVI by penalization (see (Gatheral and Jacquier 2014, Section 5.2)), in the end the SSVI approach is in fact only practically arbitrage-free, much like our NN approach, whereas it is only the GP approach that is proven arbitrage-free.

2.7 SPX Numerical Experiments

2.7.1 Experimental design

Our training set is prepared using SPX European puts with different available strikes and maturities ranging from 0.005 to 2.5 years, listed on 18th May 2019, with S_0 = \$2859.53. Each contract is listed with a bid/ask price and an implied volatility corresponding to the mid-price. The associated interest rate is constructed from US treasury yield curve and dividend yield curve rates are then

obtained from call/put parity applied to the option market prices and forward prices. We preprocess the data by removing the shortest maturity options, with $T < 0.055$, and the numerically inconsistent observations for which the gap between the listed implied volatility and the implied volatility calibrated from mid-price with our interest/dividend curves exceeds 5% of the listed implied volatility. But we do not remove arbitrable observations in the sense of violation of arbitrage relationships on training set (as assessed on the basis of discretely approximated calendar and butterfly conditions). The preprocessed training set is composed of 1720 market put prices. The testing set consists of a disjoint set of 1725 put prices.

All results for the GP method are based on using Matern $\nu = 5/2$ kernels over a $[0, 1]^2$ domain with fitted kernel standard-deviation hyper-parameter $\hat{\sigma} = 185.7611$, length-scale hyper-parameters $\hat{\theta}_k = 0.3282$ and $\hat{\theta}_T = 0.2211$, and homoscedastic noise standard deviation, $\hat{\varsigma} = 0.6876$.[3] The grid of basis functions for constructing the finite-dimensional process P^h has 100 nodes in the modified strike direction and 25 nodes in the maturity direction. The Matlab interior point convex algorithm `quadprog` is used to solve the MAP quadratic program (2.11).

Regarding the NN approach, we use a three layer architecture similar to the one based on prices (instead of implied volatilities in Section 2.3.3) in (Chataigner, Crépey, and Dixon 2020), to which we refer the reader for implementation details. We use a penalty grid Ω_h with $m = 50 \times 100$ nodes. In the moneyness and maturity coordinates, the domain of the penalty grid is $[0.005, 10] \times [0.5, 2]$.

2.7.2 Calibration results

Training times for SSVI, GP, and NNs are reported in the last row of Table 2.7.1 which, for completeness, also includes numerical results obtained by NN interpolation of the prices as per (Chataigner, Crépey, and Dixon 2020). Because price based NN results are outperformed by IV based NN results we only focus on the IV based NN in the figures that follow, referring to (Chataigner, Crépey, and Dixon 2020) for every detail on the price based NN approach. We recall that, in contrast to the SSVI and NNs which fit to mid-quotes, GPs fit to the bid-ask prices.

The GP implementation is in Matlab whereas the SSVI and NN approaches are implemented in Python. On our (large) dataset, the constrained GP has the longest training time. Training is longer for constrained SSVI than for unconstrained SSVI because of the ensuing amendments to the optimization routine. There are no arbitrage violations observed for any of the constrained methods in neither the training or the testing grid. Unconstrained methods yield 18 violations with NN and 177 with SSVI on the testing set, out of a total of 1725 testing

[3] When re-scaled back to the original input domain, the fitted length scale parameters of the 2D Matern $\nu = 5/2$ are $\hat{\theta}_k = 973.1901$ and $\hat{\theta}_T = 0.5594$.

IV RMSE (Price RMSE)	SSVI	GP	IV based NN	Price based NN	SSVI Unconstr.	GP Unconstr.	IV based NN Unconstr.	Price based NN Unconstr.
Calibr. fit on the training set	1.37% (2.574)	0.58% (0.338)	1.23% (2.897)	13.70% (9.851)	1.04% (2.691)	0.60% (0.321)	0.84% (2.163)	5.65 % (2.456)
Calibr. fit on the testing set	1.52% (2.892)	0.57% (0.355)	1.29% (2.966)	14.27% (10.347)	1.09% (2.791)	0.57% (0.477)	0.86% (2.045)	6.14% (2.888)
MC backtest	8.69% (22.826)	19.76% (74.017)	2.95% (4.989)	6.37% (11.764)	N/A	N/A	N/A	N/A
CN backtest	6.88% (33.545)	7.86% (35.270)	3.43% (11.976)	5.56% (26.785)	N/A	N/A	N/A	N/A
Comput. time (seconds)	33	856	191	185	1	16	76	229

Table 2.7.1: The IV and price RMSEs of the SSVI, GP and NN approaches. Last row: computation times (in seconds).

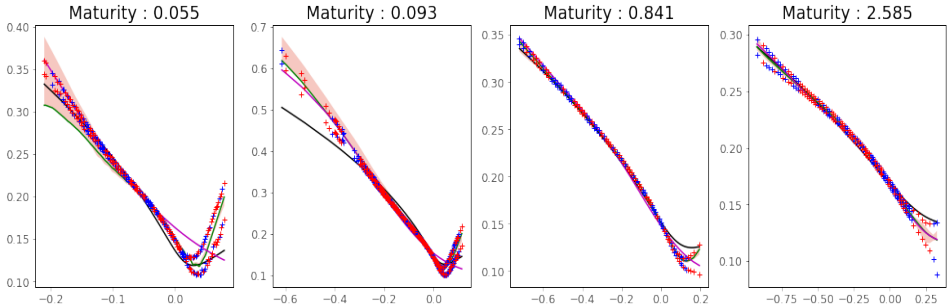
points, i.e. violations in 1.04% and 10.26% of the test nodes. The unconstrained GP approach yields constraint violations on 12.5% of the basis function nodes $\imath h$. The NN penalizations $(\text{cal}_T)^-$ and $(\text{butt}_\kappa)^-$ vanish identically on the penalty grid Ω_h in the constrained case, whereas in the unconstrained case their averages across grid nodes in Ω_h are $(\text{cal}_T)^- = 3.91 \times 10^{-6}$ and $(\text{butt}_\kappa)^- = 1.60 \times 10^{-2}$ with the IV based NN.

Fig. 2.7.1(a-b) respectively compare the fitted IV surfaces and their errors with respect to the market mid-implied volatilities, among the constrained methods. The surface is sliced at various maturities (more slices are available in the github) and the IVs corresponding to the bid-ask price quotes are also shown – the blue and red points respectively denote training and test observations.

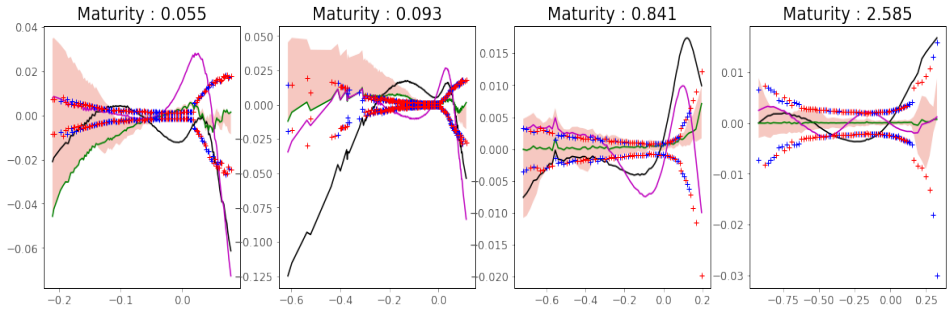
We generally observe good correspondence between the models and that each curve typically falls within the bid-ask spread, except for the shortest maturity contracts where there is some departure from the bid-ask spreads for observations with the lowest log-moneyness values. We see on Fig. 2.7.1(b) that the GP IV errors are small and mostly less than 5 volatility points, whereas NN and SSVI exhibit IV error that may exceed 15 volatility points. The green line and the red shaded envelopes respectively denote the GP MAP estimates and the posterior uncertainty bands under 100 samples per observation. The support of the posterior GP process assessed on the basis of 100 simulated paths of the GP captures the majority of bid-ask quotes. The GP MAP estimate occasionally corresponds to the boundary of the support of the posterior simulation. This indicates that the posterior truncated Gaussian distribution is heavily skewed for some points, and that the MAP estimate consequently saturates the arbitrage constraints. This indicates a tension between these constraints and the calibration requirement, which cannot be fully reconciled, most likely because some of the (short maturity) data are arbitrable (they are at least illiquid and hence noisy). See notebook for

location of arbitrages in the unconstrained approach.

Fig. 2.7.1(a-b) suggest that the data may exhibit arbitrage at the lowest maturities where the methods depart from the bid-ask spreads. This is further supported in Fig. 2.7.2(a-b) which shows the corresponding methods without the no-arbitrage constraints. In Fig. 2.7.2(a-b) we observe that the estimated IVs now fall within close proximity of the bid-ask spreads–all methods exhibit an error typically less than 5 volatility points. Note that the y-axis has been scaled for each plot in Fig. 2.7.2(b) to accommodate the wide uncertainty band of the posterior for the unconstrained GP. Whereas the uncertainty band of the constrained GP spanned at most 10 volatility points, the uncertainty band of the unconstrained GP is an order of magnitude larger, sometimes spanning more than 100 volatility points.



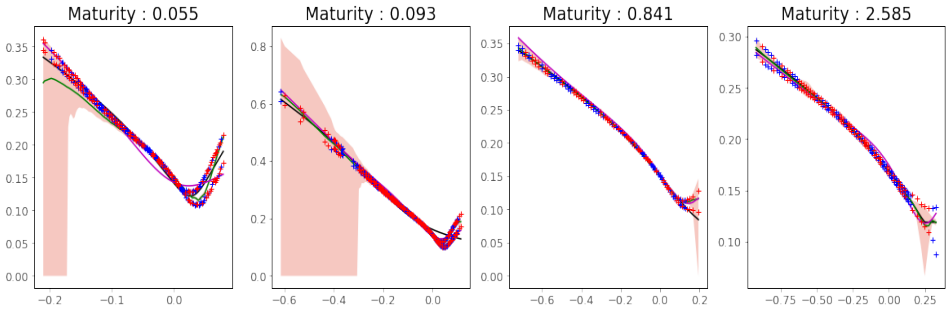
(a) Implied volatilities.



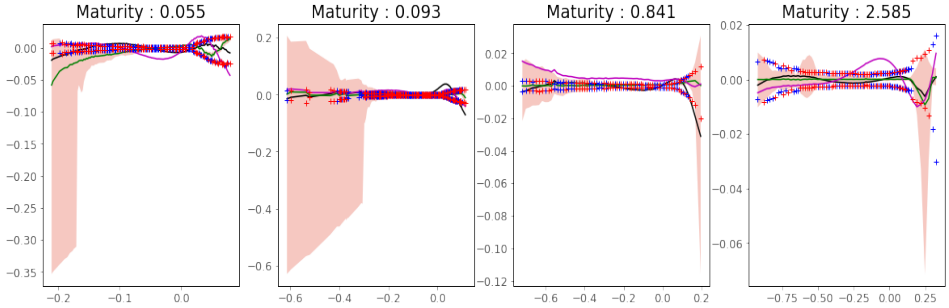
(b) Fitted IV errors with respect to mid-price IVs.

Figure 2.7.1: *Slices of constrained GP (green), NN (purple), and SSVI (black) models of SPX puts with training bid-asks IVs (+) and testing bid-asks IVs as a function of log forward moneyness (+)(the bid-ask IVs are reconstructed numerically from the corresponding bid-ask market prices). The shaded envelopes show 100 paths of the constrained GP's posterior.*

Fig. 2.7.3 shows the local volatility surfaces that stem from the three constrained approaches. Fig. 2.7.3(a) shows the spiky local volatility surface generated by SSVI, capped at the 200% level for scaling convenience. Fig. 2.7.3(b) shows

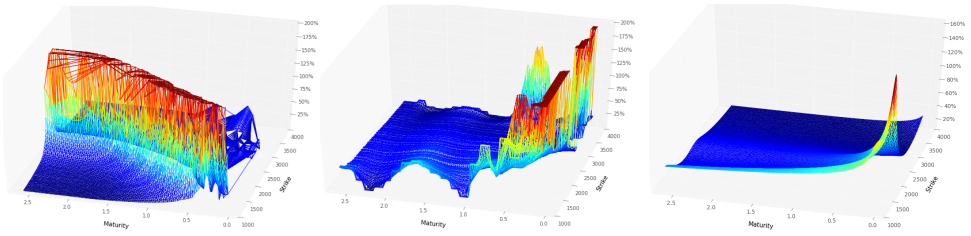


(a) Implied volatilities.



(b) Fitted IV errors with respect to mid-price IVs.

Figure 2.7.2: *Same as Figure 2.7.1 but for unconstrained GP, NN and SSVI.*



(a) The local volatility surface generated by SSVI with finite differences, capped at the 200% level.

(b) The MAP estimate of the GP local volatility surface, capped at the 200% level.

(c) The implied volatility based NN local volatility surface (with the local volatility penalization).

Figure 2.7.3: *The GP, SSVI, and NN local volatility estimate.*

the capped local volatility surface constructed from the GP MAP price estimate. Fig. 2.7.3(c) shows the (complete) NN local volatility surface.

2.7.3 In-sample and out-of-sample calibration errors

The error between the prices of the calibrated models and the market data are evaluated on both the training and the out-of-sample data set. The first two rows of Table 2.7.1 compare the in-sample and out-of-sample RMSEs of the prices and implied volatilities across the different approaches. The differences between the training and testing RMSEs are small, suggesting that all approaches are not over-fitting the training set. The GP exhibits the lowest price RMSEs.

2.7.4 Backtesting results

The first repricing backtest estimates the prices of the European options corresponding to the testing set, by Monte Carlo sampling in each calibrated local volatility model (same methodology as in (Chataigner, Crépey, and Dixon 2020, Section 7.2)). The second approach uses finite differences to price the options with the calibrated local volatility surfaces. The pricing PDEs with local volatility are discretized using a Crank-Nicolson (CN) scheme implemented on a 100×100 backtesting grid. The last two rows in Table 2.7.1 compare the resulting price backtest RMSEs across the different approaches. The NN fitted to implied volatilities exhibit significantly lower errors in the backtests, followed by NN based on prices, SSVI and GP. To quantify discretization error in these backtesting results (as opposed to the part of the error stemming from a wrong local volatility), we ran the same backtests in a Black-Scholes model with 20% volatility and the associated prices. The corresponding Monte Carlo and Crank-Nicholson backtesting IV(price) RMSEs are 2.90%(1.56) and 0.846%(4.10), confirming the significance of the above results.

2.8 Conclusion

We introduced three variations of neural net methodology to enforce no-arbitrage interpolation of European vanilla put option quotes:

1. modification of the network architecture to embed shape conditions (hard constraints).
2. use of shape penalization to favor these conditions (soft constraints).
3. additional use of local half-variance bounds in the penalization via the Dupire formula.

Our experimental results confirm that hard constraints, although providing the only fail-safe approach to no-arbitrage approximation, reduce too much the representational power of the network numerically.

Soft constraints provide much more accurate prices and implied volatilities, while only leaving space for sporadic arbitrage opportunities, which are not only occasional but also very far from the money, hence do not necessarily correspond to monetizable arbitrage opportunities once liquidity is accounted for. Once the Dupire formula is included in the penalization, the corresponding local volatility surface is also reasonably regular, at fixed day, and stable, in terms of both out-of-sample performance at fixed day and dynamically from day to day. Finally the additional penalization grid allows a more robust control of no-arbitrage conditions on a flexible domain that can be customized by the final user.

The performance of the neural network local volatility calibration method then gets close to the one of the classical Tikhonov regularization method of (Crépey 2002), but not better. However the obtention of local volatility is fast at any location once the neural net has been calibrated whereas with Tikhonov trinomial tree, local volatility needs to be estimated each time we plug a new grid. Moreover the performance are better when the neural net is calibrated with implied volatilities and the local volatility that arises from the neural net is more consistent with option quotes than with GPs or SSVI.

We thus enrich the associated machine learning literature on neural networks metamodeling of vanilla option quotes in three respects: first, by considering the associated local volatility, which is interesting both in itself and as a tool for improving the learning of the option prices in the first place; second, by working with real data; third, by systematically benchmarking our results with the help of a proven (both mathematically and numerically) classical, non machine learning calibration procedure, i.e. Tikhonov regularization, GPs or SSVI.

Chapter 3

Nowcasting network

We devise a neural network based compression/completion methodology for financial nowcasting. The latter is meant in a broad sense encompassing completion of gridded values, interpolation, or outlier detection, in the context of financial time series of curves or surfaces (also applicable in higher dimensions, at least in theory). In particular, we introduce an original architecture amenable to the treatment of data defined at variable grid nodes (by far the most common situation in financial nowcasting applications, so that PCA or classical autoencoder methods are not applicable). This is illustrated by three case studies on real data sets. First, we introduce our approach on repo curves data (with moving time-to-maturity as calendar time passes). Second, we show that our approach outperforms elementary interpolation benchmarks on an equity derivative surfaces data set (with moving time-to-maturity again). We also obtain a satisfying performance for outlier detection and surface completion. Third, we benchmark our approach against PCA on at-the-money swaption surfaces redefined at constant expiry/tenor grid nodes. Our approach is then shown to perform as well as (even if not obviously better than) the PCA (which, however, is not be applicable to the native, raw data defined on a moving time-to-expiry grid).

3.1 Introduction

In this chapter, we devise a neural network based methodology for financial nowcasting. The latter is meant in a broad sense encompassing completion of gridded values, interpolation, or outlier detection, in the context of financial time series of curves or surfaces. Toward this end we develop a generic two-step methodology, whereby a pre-processing compression stage is followed by a completion stage. Moreover, we detail two variations along this baseline, corresponding to two slightly different perspectives and significantly distinct neural network architectures.

As such our approach is not bound to vectors and matrices. For generality and notational convenience it is presented in the methodological part on arbitrary tensors (but we do not address the strong aspect of dimension reduction that typically comes with genuine tensors as opposed to matrices and vectors).

Under the so called convolutional approach, which is of the autoencoder type, we assume that the information contained in an observed tensor can be encoded into a reduced set of variables, dubbed factors. Conversely, given the factors, we can reconstruct the whole tensor with a decoder. As a limiting case, we obtain a linear, principal component analysis (PCA) kind of approach, but one itself implemented in the optimization training mode, as an autoencoder with linear activation functions (as opposed to spectral decomposition in the classical PCA case).

Under the so called functional approach, factors are rather used as a way to adjust a map taking as input a location (coordinates that may be part or not of the original tensor nodes) and returning the corresponding reconstructed value.

The convolutional approach is more particularly dedicated to completion of values on a fixed grid of coordinates, whereas the functional approach can handle moving grids, which corresponds to the vast majority of applications in financial nowcasting applications (unless the data have been transformed in a preprocessing stage to make them fit a fixed grid, entailing an undesirable layer of approximation). Moreover, in the functional approach, including additional variables is straightfoward.

The use of autoencoders as a nonlinear extension of the PCA can be traced back to the 1980s (see Chapter 14 in Bengio, Goodfellow, and Courville (2017) for a survey of autoencoder-based learning). Autoencoders have also already been used in data completion (see Kiran, Thomas, and Parakkal (2018), Strub, Gaudel, and Mary (2016)). In contrast, the neural network architecture of our functional approach is new to the best of our knowledge.

At the intersection between neural networks and finance, the related paper by Kondratyev (2018) is more about forecasting. Accordingly, we work in a mostly unsupervised setting, whereas Kondratyev (2018) is in a mostly supervised setting. Kondratyev (2018) predicts a new curve given a shock on a curve. The neural network is trained for shocks applied to a particular location. Hence, to consider a new shock, the model needs to be retrained. In contrast, our convolutional network has a latent structure capturing interdependencies between all points in the grid. This is even more obvious in the case of our functional approach, where extra variables can be provided as direct inputs to the model.

Autoencoders (hence, unsupervised learning) are also considered in Section 5.4 in Kondratyev (2018). However, this is then with a focus on curve regularization on a fixed grid, which can be done directly by decoding. The completion problem

that we are dealing in this work is more general and it requires one additional layer of numerical optimization. Moreover, Kondratyev (2018) only deals with the univariate case of curves, for which spatial regularity is a much less challenging issue.

The chapter is outlined as follows. Sections 3.2 and 3.3 introduce the problems and models. By the latter, we mean different algorithmic strategies and neural network architectures that can be used for addressing the former. Section 3.4 lays an experimental setup putting the different models on comparable grounds. Sections 3.5, 3.6 and 3.7 present repo curves, equity derivative implied volatility surfaces and at-the-money swaption implied volatility surfaces case studies on real data sets. Section 3.8 concludes and discusses further research perspectives in connection with the quantitative finance and machine learning literatures.

Any notation of the form $\min_x \Lambda(x, y)$ means that we minimize in x a loss Λ given the value y of additional parameters; $x^\star$ then refers to a numerical minimizer of $\Lambda(x, y)$ (which is typically nonconvex in x), for this given y.

3.2 Problems

We consider a data set consisting of a time series of observations, each consisting of m points, or features, structured as a multivariate tensor. By the latter, we mean a discretized cube (curve or surface in our case studies, but the methodology is generic) of values of homogenous quantities, such as rates of different terms, implied volatilities of different strikes and maturities, etc., defined at each tensor grid node.

3.2.1 Compression

The compression problem is mainly a pre-processing stage that aims at reducing the dimensionality m of a feature space, i.e. the number of grid nodes in each tensor (here assumed constant across observations ω, see Section 3.3.3 regarding the variant of the functional approach with a possibly variable m_ω). Assume that each observation takes its values in (a subset of) $\mathbb{R}^m$. We call encoder E any injective map from a relevant subset $\mathcal{S}$ of $\mathbb{R}^m$ to a space $\mathbb{R}^f$ of factors, where $f \ll m$ is the number of factors. Conversely, one would like to be able to reconstruct the m values of a tensor from any set of factors, or code, thanks to a map, called decoder, $D : \mathbb{R}^f \to \mathcal{S}$. The compression challenge is to build D and E such that $D \circ E : \mathcal{S} \to \mathcal{S}$ is bijective and "as close as possible to identity" (cf. Bengio, Goodfellow, and Courville (2017, Chapter 14)).

The inspection of common financial time series of tensors suggests that, in their case, this challenge is somehow not unreasonable. Indeed, structural constraints

often exist between the values at different tensor nodes, e.g. arbitrage pricing relationships throughout the option chain. Moreover, usual financial tensors exhibit some spatial regularity, in the sense that values at grid nodes vary smoothly with respect to node location (think of interest rates with respect to their term or implied volatilities with respect to the maturity and strike of an option). In addition, some coordinates may have a regularizing effect. For instance, in the region of large expiries, the at-the-money swaption implied volatility surface is mostly affected by translation moves (and not so much by steepening, etc.) as time passes (see Section 3.7). Last, some (monotonicity, convexity,...) patterns are often apparent (e.g. the well-known volatility smile in equity derivative, and some similar features in interest rate swaption implied volatility surfaces, cf. Figure 3.7.1).

Both maps E and D are sought within classes of neural networks with respective parameters ε and δ, collectively denoted by θ. The motivation for using neural networks in this context is their nonparametric (or, at least, very expressive) and nonlinear features. Gaussian processes for instance would be much less flexible, with only a few, e.g. two, kernel hyperparameters for squared exponential kernel to calibrate a full data set of thousands of tensors.

We include into θ weights, biases, as well as any variable calibrated during the compression stage. Denoting $E = E_\varepsilon$ and $D = D_\delta$ in reference to this parameterization, the compression stage is the training of the neural networks according to the following optimization problem:

$$\min_{\theta=(\delta,\varepsilon)} \sum_{\omega\in\Omega} \sum_{(n,y)\in\omega} \Big(y - \Big(D_\delta\big(E_\varepsilon(\omega)\big)\Big)_n \Big)^P, \tag{3.1}$$

where Ω stands for the training data set (cf. Section 3.4).

Certain additional properties are desirable for D and E. The parameterization θ should allow for a robust and fast numerical solution to the problem (3.1). This may be harder to achieve for some deep neural networks too sensitive to the initialization of their parameters. In particular, two similar tensors should give rise to similar codes and vice versa, i.e. we want D and E to be "sufficiently smooth" in such way as to preserve distance in the subspace.

3.2.2 Completion

Having found a parameterization $\theta^\star = (\delta^\star, \varepsilon^\star)$ that ensures a satisfying reconstruction loss in (3.1), the completion task consists in the exploitation of $D_{\delta^\star}$ in order to find the missing values of an incomplete observation ω (of the current day, say, to be completed based on the complete observations of the previous days, used as training set).

Toward this end, we introduce the following optimization problem:

$$\min_{c} \sum_{(n,y)\in\omega} \Big(y - \big(D_\delta(c)\big)_n\Big)^P, \tag{3.2}$$

considered for $\delta = \delta^\star$. The completed tensor is then defined as the image $D_{\delta^\star}(c^\star)$ of the code $c^\star$ by the decoder $D_{\delta^\star}$. Obviously, the more missing values, the harder the completion task (higher overfitting risk, unless some appropriate regularization is used).

Note that, thanks to the compression step, the number of variables to estimate is drastically reduced in (3.2), to some reference number, i.e. the dimensionality of c (e.g. 4, 15, or 8 in our repo, equity index derivative and interest-rate swaption case studies), independent of the number of unknowns in the native, "uncompressed" completion problem (such as the number of missing implied volatility values in a to-be-completed surface). Moreover, a factorial representation with $f \ll m$ filters out the unlikely tensors (as outlined by the reconstruction error from our neural networks, cf. Section 3.2.3) that could otherwise arise from a decoding due to the ill-posedness of large-scale arg-minimization problems. The regularity of the map $D_{\delta^\star}$ can sometimes be exploited to ease the completion, by initializing the numerical solution of (3.2) with the encoding of the last fully observed (e.g. already completed) tensor.

Literature Review The literature on completion primarily deals with data structured on a fixed grid. This means that columns in the data set refer to the same feature (in our case: financial instrument). This is not consistent with most financial nowcasting applications, for which, in particular, the time-to-maturity decreases with calendar time. To the best of our knowledge, only naive interpolation methods on a given tensor, without possible exploitation of a data set, are available in the case of a moving grid.

The standard completion framework relies on a low rank representation of the data set (see Nguyen, Kim, and Shim (2019)). Along this line (but on a possibly moving grid), we compress each observation in a code which can be seen as a latent vector. However, in contrast with methods such as SVD, alternating least squares (see Hastie, Mazumder, Lee, and Zadeh (2015)), or denoising autoencoders (see Strub and Mary (2015)), which learn a user matrix, we do not consider the interaction between the observations (i.e. the dynamics): at this stage at least, we focus on the interaction between the variables (instruments).

Finally, standard completion methods in recommender systems assume missing completely at random (MCAR) values dispersed throughout the whole data set. In our case studies missing values are located completely at random but only for the current observation.

3.2.3 Outlier Detection

Hawkins (1980) defines outliers as "observation which deviates so much from other observations as to arouse suspicion it was generated by a different mechanism". Outlier detection is of course a crucial issue in finance. For instance, investment banks receive market information from a data provider. Sometimes, the data can be polluted with errors of various sources, or "different mechanism", whether it is data feed bugs, fat finger of other market participants, or failure from computation processes (for instance, implicitation of volatility surface from option prices). It can be either a punctual outlier, i.e. a single value of the tensor is too far away from what it should be, or the whole tensor may have a shape that is very unlikely.

To detect the punctual outliers, many simple methods are available, based on smoothness metrics or on historical percentile ranges of the values. To detect shape outliers, some criteria can be checked for very specific data sets, e.g. non-arbitrage butterfly/calendar spread conditions in the case of option prices.

Here we propose a general method to detect both punctual outliers and shape aberration. The functional variant of our method works on an unstructured grid.

To say that the tensors generated from the "normal mechanism" is of a certain form is equivalent to say that the mechanism generates values that lie in a sub-manifold $\mathcal{S}$ of the initial feature space (cf. Section 3.2.1). Finding this sub-manifold is equivalent to detecting anomalies. From this point of view, anomaly detection and compression/decompression are two sides of the same coin. Indeed, from an information theory point of view, there is an equivalence between being an anomaly and being hard to reconstruct (large reconstruction error in a lossy data compression setup, low or even negative compression rate in a lossless data compression setup): See the seminal paper by Shannon (1948) or Chapter 4 in MacKay and Mac Kay (2003). That is, a compression/decompression setup provides a natural anomaly detection tool.

Specifically, we identify an outlier as an observation whose reconstruction error (cf. (3.1)) is above a predefined threshold.

Some key practical questions in outlier detection are how a threshold for outlier detection should be chosen or how one can validate the method. In principle this can only be addressed by human expertise. An expert would gradually diminish the threshold until the newly detected 'outliers' are no longer considered such by the expert. The method is valid and performs well if the outlier detection in a validation set is consistent with the expert view (so that, in particular, the threshold is stable through time and does not need to be reassessed too frequently).

However, our compression methodology also provides a validation tool for the quality of our outlier detection method. Namely, one can corrupt some of the data (manually or in an automated fashion) and check whether the outlier detection procedure identifies the corrupted data.

Our approach also provides guidance to a human expert for anomaly correction. Currently experts only rely on naive heuristics, such as interpolation between different points of a surface, who cannot automatically exploit the overall data set of surfaces. In the outlier detection validation framework of the previous paragraph, one can also check whether the correction that our approach provides is closer to the true data than to the corrupted ones.

Literature Review Among many related references on outlier detection:

- Patcha and Park (2007), Chandola, Banerjee, and Kumar (2009), Omar, Ngadi, and Jebur (2013), or Anandakrishnan, Kumar, Statnikov, Faruquie, and Xu (2018) provide surveys, the last one specialized in finance and the next-to-last one on machine learning techniques;
- Lakhina, Joseph, and Verma (2010) use PCA, An and Cho (2015) variational autoencoders, Schlegl, Seeböck, Waldstein, Langs, and Schmidt-Erfurth (2019) generative adversarial networks, Lakhina, Joseph, and Verma (2010) and Cappozzo, Greselin, and Murphy (2020) semi-supervised learning. Chaloner and Brant (1988) and Cansado and Soto (2008) resort to Bayesian methodologies;
- Ro, Zou, Wang, and Yin (2015) is about high-dimensional data, Anandakrishnan, Kumar, Statnikov, Faruquie, and Xu (2018) about high dimensional big data, Rocke and Woodruff (1996) about multivariate data, Goix, Sabourin, and Clémençon (2017) and Goix, Sabourin, and Clémençon (2015) about detection of anomalies among extremes.

3.3 Models

The main innovation of this work is the functional approach. The description of the PCA and linear projection methods are mainly provided so that the reader can compare carefully both frameworks. The PCA and linear projection methods are also used for benchmarking purposes in the swaption case study of Section 3.7 for which both approaches are available. The base PCA method is of course standard. The linear projection variant of it is detailed in Section 3.3.2. The description is short because the method falls directly under the umbrella of sections 3.2.1 and 3.2.2 (as opposed to the functional approach of section 3.3.3, which requires a specific development).

3.3.1 The Convolutional (Autoencoder) Approach

Typical autoencoder architectures are composed of two successive feedforward neural networks E and D, the encoder and the decoder. Both networks can be constituted of several layers, intermediated by nonlinear activation functions, with an overall bottleneck structure (to enforce compression in the middle).

Convolutional layers have been introduced for image processing and, more generally, any data structure represented as a tensor. These networks aim to model the interactions between close points (whereas dense layers bind any output unit to all input units). Spatial regularity properties are handled by a convolutional structure of the neural network architectures, whereby the only (non-zero) connections are between units corresponding to adjacent (in a suitable sense) grid nodes (cf. Figure 3.7.3). The network then also uses fewer parameters, which reduces the complexity of the corresponding compression problem. For implementation details such as kernels and padding, we refer to Chapter 9 in Bengio, Goodfellow, and Courville (2017).

3.3.2 The Linear Projection Approach

It is well known that an autoencoder with linear activation functions and an L_2 reconstruction error is equivalent to a PCA (see Chapter 14 in Bengio, Goodfellow, and Courville (2017)). As a limiting case of the above, we consider a linear, PCA kind of benchmark, but one itself implemented as an autoencoder with linear activation functions (as opposed to spectral decomposition for classical PCA implementation). With respect to classical PCA (which will also be included in our case studies), this approach involves an additional bias parameter. Moreover, it allows benefiting from the implicit regularization provided by early stopping in the related training procedure (see Section 3.4), as opposed to a regularization provided by truncation of the lowest eigenvalues in spectral decomposition based PCA implementation.

3.3.3 The Functional Approach

We introduce a variant of the above, especially suited to interpolation purposes (without reference to a fixed grid of nodes). This approach relies on a parameterized function $D = D_\delta(c, n)$ of a code c and a node location n, where the latter no longer needs belong to a pre-determined grid. Here δ corresponds to the parameters of the decoder D, whereas the approach does not entail any encoder (at least, not explicitly).

The compression is written as (compare with (3.1), using a similar notation as

well as $C = (C_\omega)_{\omega\in\Omega}$)

$$\min_{\delta,C} \sum_{\omega\in\Omega} \sum_{(n,y)\in\omega} \Big(y - D_\delta\big(C_\omega, n\big)\Big)^P. \tag{3.3}$$

Then, given a single, possibly partial observation ω, the completion is given as (similar to (3.2))

$$\min_{c} \sum_{(n,y)\in\omega} \Big(y - D_\delta\big(c, n\big)\Big)^P, \tag{3.4}$$

considered for $\delta = \delta^\star$. Importantly, for each given δ, the minimization (3.3) decouples into one (full observation) minimization (3.4) for each $\omega \in \Omega$. Hence, the larger compression problem (3.3) can be solved numerically as a succession of smaller problems (3.4), in conjunction with gradient iterations in the direction of δ. This ensures the scalability of the approach. It also makes it amenable to online learning. The above observation also shows the consistency between (3.3) and (3.4) in the sense that, if a full observation ω is used in (3.4), it should yield $c^\star = C^\star_\omega$ (assuming global and unique minima to all problems for the sake of the argument).

Under this approach, dubbed functional, the decoder takes as input the location n of the point, in addition to the factors c (see Figure 3.5.1). It rebuilds each point individually, as per $n \to D_\delta(c, n)$. The network is thus able to interpolate between the nodes of the data grid. The concept of neighborhood intervenes through the argument n of D, but the parameterization δ as well as the code c are common to all locations n. The compression (3.3) can also accommodate incomplete data or discretization changes, i.e. varying grids in the training data. This feature allows training the functional network with "missing completely at random data" (MCAR, in the statistical missing data terminology).

By comparison, under the convolutional approach of Section 3.3.1, the concept of neighborhood intervenes through $\theta = (\delta, \varepsilon)$, since each point of the grid is only sensitive to a subset of connections (the convolutional architecture only connects neighbouring points, cf. Figure 3.7.3). The encoding c is obtained directly thanks to E, when the observation is complete, or by numerical completion (as always under the functional approach) otherwise.

3.3.4 Synthesis

To conclude this section, Tables 3.3.1 and 3.3.2 summarize and put into perspective the different approaches referred to in the above.

Also note that, from a numerical complexity point of view, the functional approach is less sensitive to the dimension than, say, a classical autoencoder on

Encoder	Implicit and non-linear $\hat{c} = \arg\min_{c} \sum_{(n,y)\in\omega} \Big(y - D_\delta\big(c,n\big)\Big)^P$
Decoder	Analytic and non-linear $\hat{y} = D_\delta\big(c,n\big)$
Compression (training) step	Optimization w.r.t. (δ, c) $\min_{\delta,C} \sum_{\omega\in\Omega} \sum_{(n,y)\in\omega} \Big(y - D_\delta\big(C_\omega,n\big)\Big)^P$
Reconstructed surface Reconstruction	Implicit $\hat{y} = D\left(\arg\min_{c} \sum_{(n,y)\in\omega} \Big(y - D_\delta\big(c,n\big)\Big)^P, n\right)$
Completed surface	$\hat{y} = D\left(\arg\min_{c} \sum_{(n,y)\in\omega} \Big(y - D_\delta\big(c,n\big)\Big)^P, n\right)$

Table 3.3.1: The functional approach.

	PCA	Convolutional
Encoder	Analytic and Linear $\hat{c} = E_\varepsilon(y)$	Analytic and non-linear $\hat{c} = E_\varepsilon(y)$
Decoder	Analytic and linear $\hat{y} = D_\delta\big(c\big)$	Analytic and non-linear $\hat{y} = D_\delta\big(c\big)$
Compression (training) step	Optimization w.r.t. (δ, ε) $\min_{\theta=(\delta,\varepsilon)} \sum_{\omega\in\Omega} \sum_{(n,y)\in\omega} \Big(y - \Big(D_\delta\big(E_\varepsilon(\omega)\big)\Big)_n\Big)^P$	Optimization w.r.t. (δ, ε) $\min_{\theta=(\delta,\varepsilon)} \sum_{\omega\in\Omega} \sum_{(n,y)\in\omega} \Big(y - \Big(D_\delta\big(E_\varepsilon(\omega)\big)\Big)_n\Big)^P$
Reconstructed surface Reconstruction	Explicit/analytic $\hat{y} = D\left(E(y)\right)$	Explicit/analytic $\hat{y} = D\left(E(y)\right)$
Completed surface	$\hat{y} = D\left(\arg\min_{c} \sum_{(n,y)\in\omega} \Big(y - D_\delta\big(c\big)\Big)^P\right)$	$\hat{y} = D\left(\arg\min_{c} \sum_{(n,y)\in\omega} \Big(y - D_\delta\big(c\big)\Big)^P\right)$

Table 3.3.2: PCA and convolutional approaches.

a fixed grid (including our convolutional approach), for which the size of the grid typically grows exponentially with the dimension.

3.4 Experimental Methodology and Setting

The data and code for the equity and repo case studies can be found on a public github repository `https://github.com/mChataign/smileCompletion` (the data and code for the swaption case study are proprietary).

In this section, we devise an experimental methodology and the learning procedures, so that all models are set on comparable grounds.

All the optimization (compression or completion) problems are solved with the Adam adaptive learning rate stochastic gradient algorithms of Kingma and Ba (2015). The output of a neural network is by construction non-convex with respect to its parameters. So are therefore all our loss functions. The Adam algorithm has proven its robustness in non-convex optimization context. With the help of automatic adjoint differentiation, it provides fast training for most neural networks architectures. However, no convergence is guaranteed theoretically.

For the compression stage, we make a $80 : 20$ split of a full data set into a training set and a test set. The split is chronological in order to avoid look-ahead bias (cf. Ruf and Wang (2020)). The training set is further split into a calibration and a validation data set. The former is used for computing the gradients driving the numerical optimization in the training problem, whereas the latter is used for determining an early stopping rule that provides implicit regularization, as detailed below.

The learning rate of the Adam optimizer is set to 0.001. Mini-batch learning is used in the repo and equity index derivative case studies, whereas batch-learning is employed with swaption volatilities. The gradient descent is driven by the loss computed on the calibration set, but the validation error is the loss function computed on the validation data set. The learning procedure is stopped when we do not observe any decrease of the validation error during a certain number of iterations, called patience. The parameterization returned by the compression is the one that minimizes the validation error. Early stopping in this sense limits the generalization error (cf. Engl, Hanke, and Neubauer (1996)), i.e. the gap between the reconstruction errors computed on the calibration data set and a new, unobserved data set, the role of which is played by the test set. Sometimes, as detailed later, a penalization term is added to the compression loss function in order to provide a more regular and stable minimization. A maximum number of iterations is fixed to 10^4 at compression stage and 10^3 at completion stage, in order to cap the length of the optimizations.

All approaches are implemented in Python, using the tensorflow package in the swaption case study and pytorch in the two others. Note that all hyperparameters are chosen manually, rather than by grid search or random search techniques. Grid search is not possible because we have too many hyperparameters. Exploring different neural net architectures would be too demanding computationally. How-

ever, some of the hyperparameters can be fixed based on human expertise. For instance, 15 factors in our case study of Section 3.6 is the number of factors that equity derivative traders commonly use in PCAs (after interpolation on a fixed grid, as they are faced with moving grids).

3.4.1 Performance Metrics

We want to assess, for each approach, the performance of the corresponding compression and completion procedures, as well as the behavior (distribution and dynamics) of the resulting factors. For the compression, we consider the average root mean square reconstruction error $RMSE_\omega$ on the test set Ω' (root mean square error $RMSE_\omega$ between the values at the nodes of the tensor ω and their reconstructed counterparts, i.e.

$$\sqrt{\frac{1}{m}\sum_{(n,y)\in\omega}\left(y-\left(D_{\delta^\star}\left(E_{\varepsilon^\star}(\omega)\right)\right)_n\right)^2} \tag{3.5}$$

(or the analogous quantities with m_ω and $D_{\delta^\star}\left(C^\star_\omega, n\right)$ instead of m and $\left(D_{\delta^\star}\left(E_{\varepsilon^\star}(\omega)\right)\right)_n$, as relevant). In the case of the functional approach the encoder E is implicit and its definition is detailed in table 3.3.1. We refer to (3.5) as the reconstruction loss in the compression stage of our case studies given that ω is a complete surface.

In constrast with (3.5),

$$\sqrt{\frac{1}{m}\sum_{(n,y)\in\omega}\left(y-\left(D_{\delta^\star}\left(\hat{c}\right)\right)_n\right)^2} \tag{3.6}$$

(or m_ω rather than m in the case of the functional approach) is called the completion loss when we compare the complete original observation with the completed observation. This completed observation is given by the decoder for code values $\hat{c}$ which are calibrated on the incomplete view provided by ω.

In the case of interpolation benchmarks, there is no compression stage and no code is involved at the completion stage: the completion loss is then defined by the RMSE between the interpolated surface (from an incomplete ω) and the original complete ω.

We provide a focus on the observation ω leading to the worst $RMSE_\omega$ over the test set, in order to identify the locations that are less well handled (e.g. short option maturities). In addition, we display the time series of the codes. A good compression should exploit each factor in the code (we should not observe factors stuck at zero).

The quality of the completion is assessed by a backtest on the test set. Each day of Ω', we solve the problem (3.2) or (3.4), initialiazing the factors with the fully informed encoding of the previous day. We then mask 90 % of the points in each tensor of the test set. For each such observation $\omega \in \Omega'$, we check the reconstruction $RMSE_\omega$ between the completed surface and the true one. Like for compression, we plot the worst completion obtained on the test set Ω'.

3.4.2 Introduction to the Case Studies

We provide numerical results on three daily time series of real financial data: repurchase agreement yield rates, equity implied volatility surfaces and at-the-money swaption implied volatilities. However, the swaption implied volatilities have been preprocessed by our data provider to fit a fixed grid (whereas the native, raw data had a moving time-to-expiry). A preprocessing entails an unquantifiable bias and our recommendation would be to apply the functional approach to the original data (whenever available). The main motivation for the third example is that one can then benchmark the functional approach against PCA and the convolutional approach.

The advantage of working with yield rates or implied volatilities, instead of the corresponding option prices, is that these are scaled quantities, exempt from first order dependence on contract characteristics such as nominal, time-to-maturity, actual level of the underlying in at-the-money option data, etc., which should otherwise be added to the set of explanatory variables in all learning procedures. The ensuing arbitrage issue is discussed in the next subsection.

3.4.3 Discussion of the Arbitrage Issue

Arbitrage constraints can be expressed naturally in terms of options prices using calendar spread and butterfly. But in terms of implied volatility, they are non-trivial, even in the simplest case of equity derivatives (for which they are fully stated in Roper (2010)). No compression/completion method applied to implied volatility surfaces provides a way to deal with those constraints without coming back inherently to option prices. In order to circumvent that problem, one could apply our approach to the coefficients of a (e.g. local vol) model, from which non arbitrable prices and implied volatilities could be derived in a second step. However, we do not choose this route because:

- the market practitioners, who play both the roles of human experts and users, have built intuitions over decades on implied volatilities. They think of option prices directly in terms of implied volatilities. Providing them with

a good recommendation tool in terms of a quantity that is familiar to them is of great value and the primary purpose of our approach;

- most of the times, the starting point for calibrating a model (e.g. Dupire) is nothing else than the implied volatilities. Therefore the trader must correct the anomalies *before* the implied volatility surface can be plugged as an input to model calibration. Hence one of the requirements of our proposed approach is that it should be model-free;

- Having said this, if one assumes that, on the one hand, most of the surfaces in our database are arbitrage-free and, on the other hand, a more regular surface is less prone to arbitrage opportunities, then one concludes that our model should tend to remove part of the arbitrages present in the data. This can actually be seen empirically on some of the examples in Section 3.6. This is a natural by-product of anomaly correction and it also eases the calibration process.

Similar comments apply on most markets (beyond equity implied volatility), including the ones of our three case studies, i.e. repo contracts, handled by traders in terms of yield curves, and equity index derivatives and swaptions, which are handled in terms of implied volatility.

3.5 Repo Curves

Our first case study bears on the nowcasting of repo rates, based on an 2013–2019 daily time series of repo yield curves (repo rates, where repo is a shorthand for repurchase agreement).

The grid of nodes in the data is unstructured, in the sense that the corresponding dates (time-to-maturities of bonds with standardized maturity dates) vary, in both number and location, from day to day (with as little as two or three points on particularly idle days), see e.g. Figure 3.5.2. Indeed, as the expiration dates used to compute the repo curve are fixed, and the variable of interest for the repo curve shape is rather time to expiry, the latter decreases as the expiry date approaches. For a given repo curve, the times to expiry for which the repo value is available is not known in advance for that reason. Therefore, there is no canonical way to have a systematic representation of repo curves on a fixed grid, one would need to introduce artificial time to expiry of interest and interpolate/extrapolate (which poses issues of its own) the repo curve to get the values, and then working on transformed data. This is the situation the functional approach is tailored for. By not making any assumptions on the domain of input (time to expiry), the functional approach enables to handle unaltered data, by treating the time-to-maturity of a transaction as an input value (cf. Figure 3.5.1).

3.5.1 Functional Network Architecture

Our functional approach is implemented by a single feed-forward neural network composed of three fully-connected layers with 20, 20 and 1 units (see Figure 3.5.1). Hyperbolic tangent activation is applied to each but the output layer for the same

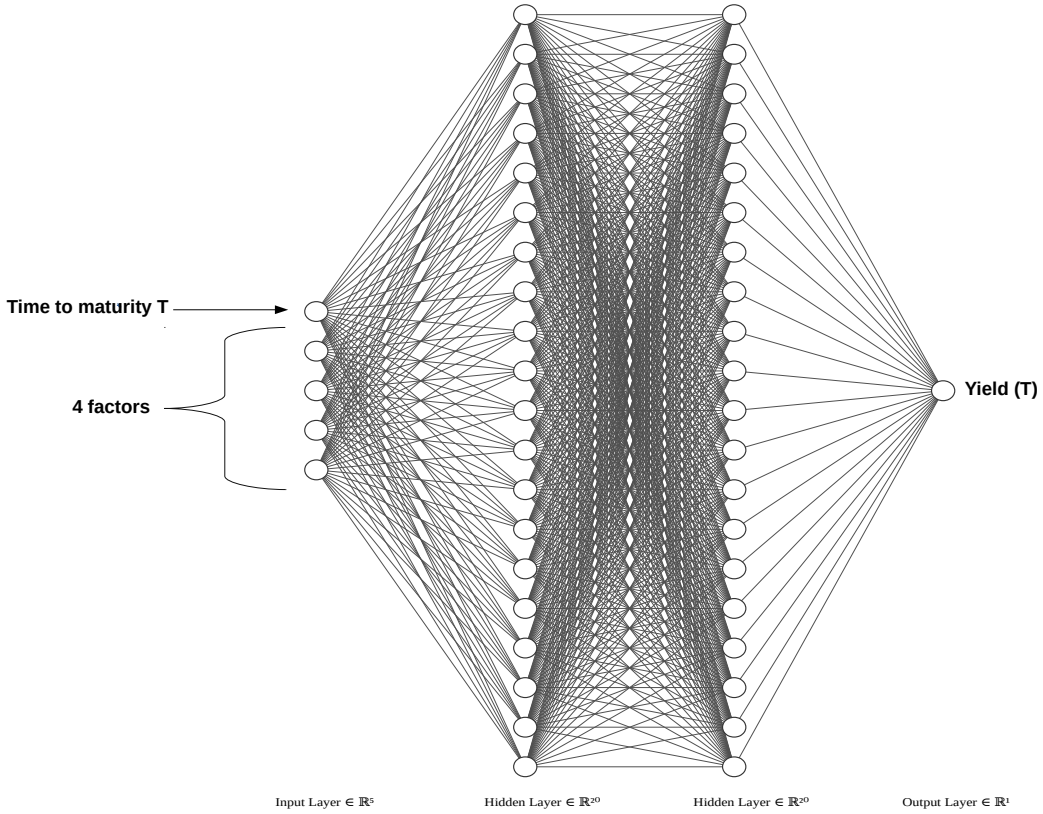


Figure 3.5.1: Network of the functional approach used in the repo case study. Here and in Figures 3.6.1 and 3.7.2 below, the graphs have been produced using the style FCNN of the NN-SVG software: the units and the connections between them are represented by circles and edges.

reasons as above (and the output layer is linear).

3.5.2 Numerical Results

As the bottom panels of Figure 3.5.2 illustrate, the parameterization is flexible and can accomodate different curve shapes or node localizations.

As explained in Section 3.2.3, the compression stage can be used for detecting an abnormal curve and correcting it with a more likely one. The distinction between inliers and outliers is determined by a threshold on the reconstruction error.

A bad reconstruction is taken as a signal that the codebook is not able to explain the corresponding observation. We then conclude that the latter does not lie in the manifold $\mathcal{S}$ of the "usual" curves, hence we classify it as an outlier (see Section 3.2.3). We can then correct (replace) these data by the curve reconstructed from the decoder with the factors calibrated on the current values, i.e. by the output of the corresponding completion (3.4).

The lower panels of Figure 3.5.2 show the gap between the observed data points and the reconstructed ones. The upper left panel spots the outliers at a 0.035 absolute RMSE threshold. The upper right panel gives an example of outlier correction.

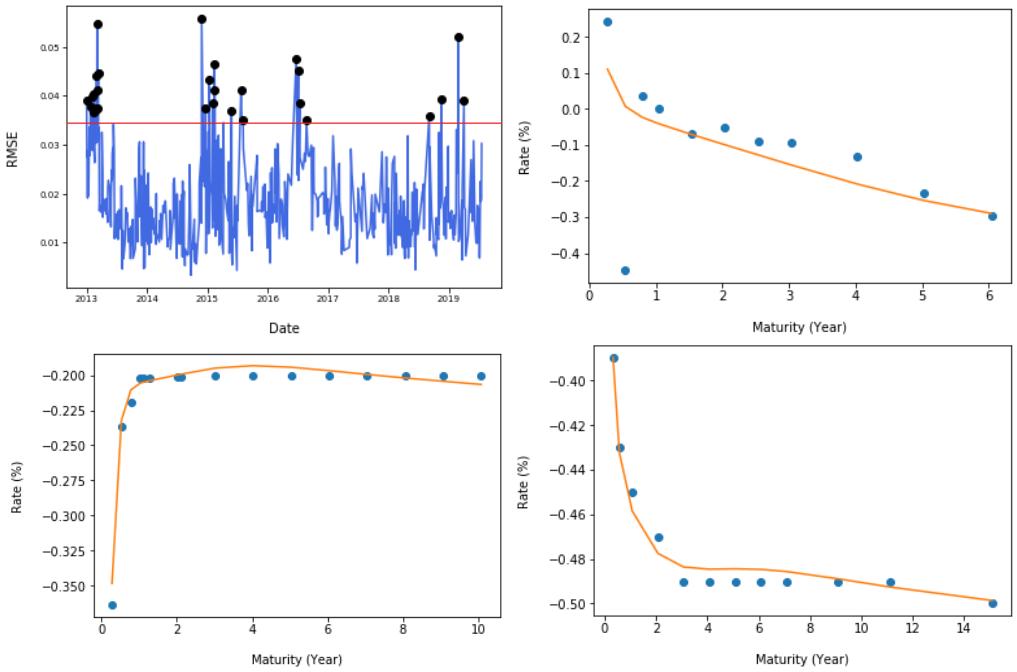


Figure 3.5.2: *(Bottom)* Interpolation of two inlier repo curves; *(Top left)* Time series of the (absolute) RMSEs on the repo data and 0.035 RMSE threshold; the spotted values correspond to the outliers at the chosen threshold. *(Top right)* Interpolation of an outlier repo curve.

3.6 Equity Derivative Implied Volatility Surfaces

As a second experiment, we apply our functional approach to Black–Scholes implied volatilities surfaces of equity index derivatives. The corresponding volatilities price options on the Nikkei 225 index from 2015 to 2018 (included), corresponding

to 1544 observable surfaces. The order of magnitude of implied volatilities fluctuates between 0.15 and 1.2. We include the forward rate as an exogenous variable that can be plugged into the functional network (3.5.1) along with log-maturity and log-moneyness.

As in the repo case study, the grid of nodes in the data is unstructured, in the sense that the corresponding dates (time-to-maturities of equity index options with standardized maturity dates) vary over time. But, again, this is the situation the functional approach is tailored for (cf. Figure 3.5.1). The corresponding architecture of the functional approach is then similar to the one used for repo curves in the previous section, except that the log-time-to-maturity and the log-moneyness are used as the (two dimensional) localization inputs, and that 15 latent variables are used as input for the decoder for encoding each volatility surface (instead of only 4 previously): see Figure 3.6.1. Moreover, one can also easily incorporate the forwards as exogenous variables. For taking them into account, it suffices to add to the network of Figure 3.6.1 an additional feature (input unit) containing the level of the forward swap rate with maturity T. Hence, the units for the maturity T indicate the common location of the corresponding volatilities and forward rates.

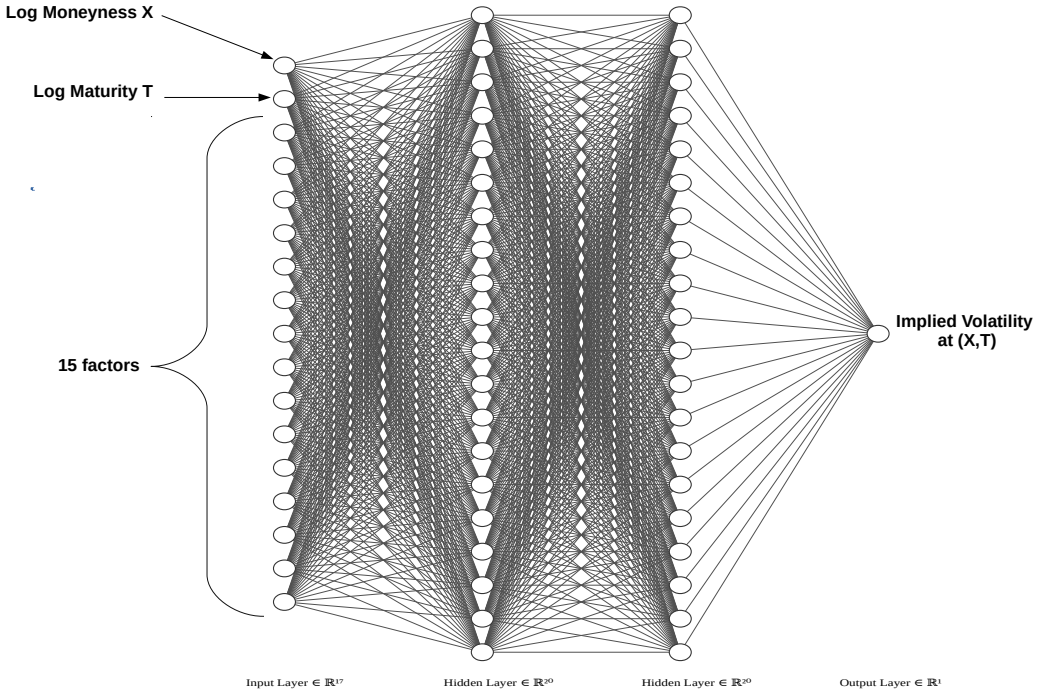


Figure 3.6.1: Network of the functional approach used in the equity case study (style FCNN of the NN-SVG software, cf. Figure 3.5.1).

3.6.1 Compression

We first calibrate our functional approach with the compression stage. Toward this end, we execute the optimization (3.1) on the training set and then calibrate codes with (3.2) for each observation in both testing and training data sets. The quality of the compression is assessed through the reconstruction errors reported in Table 3.6.1. By reconstruction error we mean the gap between the original surface and the surface induced by the code calibrated from (3.2).

We emphasize the difference between a reconstucted surface (as above) and a completed surface (considered later): the code leading to the completed surface is calibrated from an incomplete surface whereas the one for the reconstructed surface is obtained from a complete real surface.

	Functional	Functional with Forward
Training set	0.0070	0.0063
Testing set	0.0058	0.0064

Table 3.6.1: RMSEs for reconstructed implied volatilities.

In all four cases, the RMSEs in Table 3.6.1 are very small compared to the order of magnitude of implied volatilities (between 0.15 and 1.2). The results show no sign of overfitting (the reconstructions error are similar on the training set and the testing set). Moreover the comparison between the two columns of the table indicates that there is no benefit in including the forward price as an exogenous variable in our network.

Another way to assess the performance of the compression stage is to consider the worst compression, i.e. the surface yielding the highest reconstruction error. This worst reconstruction corresponds to a RMSE of 0.0096. It is represented in Figure 3.6.2, with the real surface on the top-left corner, the reconstructed couterpart on the top-right corner and the pointwise absolute difference between the two at the bottom.

We notice that the errors are concentrated on the upper tail (deep in the money call options) and for short maturities, which corresponds to illiquid options.

A bad reconstruction of a surface can also be used for qualifying it as an outlier. For instance, Figure 3.6.3 shows the implied volatility values corresponding to the most extreme strikes in Figure 3.6.2: original data points as dots and curves from the reconstructed surface. The left panel corresponding to the illiquid upper tail shows around the maturity 1.5 year a very low point that an expert would indeed qualify as an anomaly. The correction (i.e. the reconstructed surface) ignores this anomaly and has a more reasonable shape from a practitioner of view.

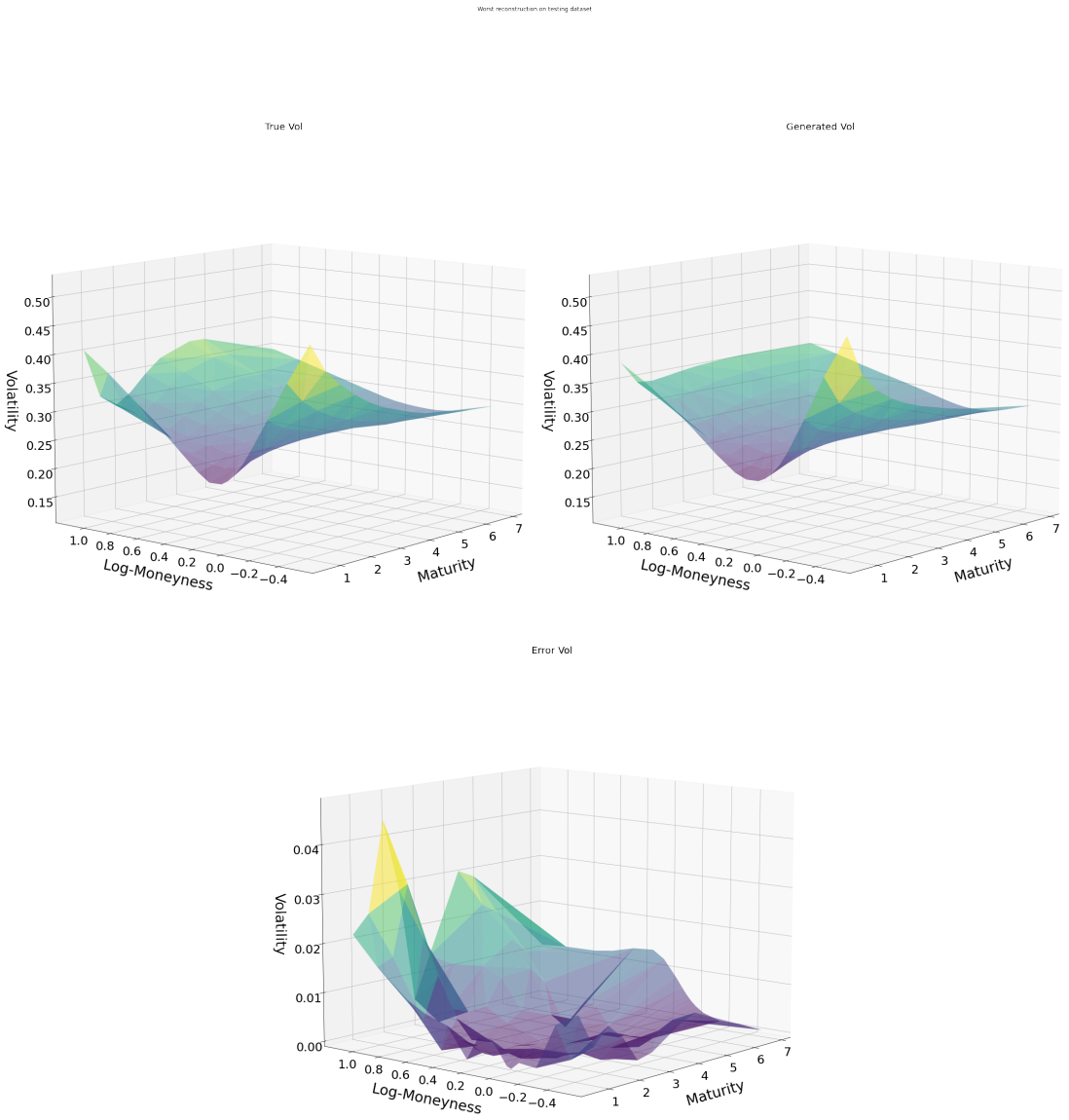


Figure 3.6.2: Original surface vs compressed surface yielding worst RMSE.

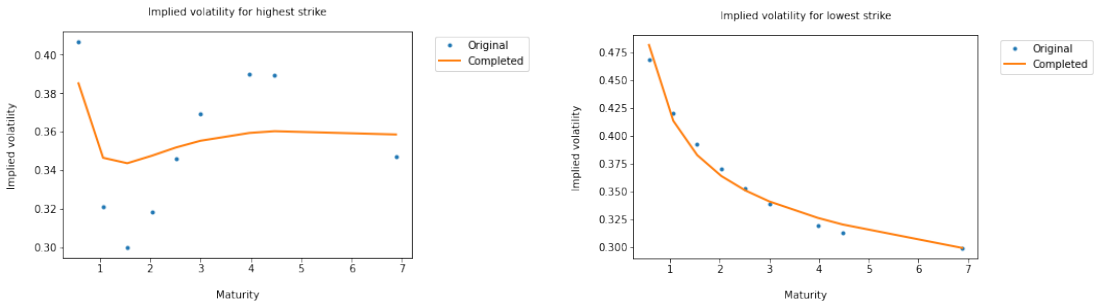


Figure 3.6.3: Tails of compressed surface vs original implied volatilities.

The left part of Figure 3.6.5 shows that the corrected surface is not prone to calendar arbitrage: the sensitivity to the maturity of the corresponding implied total variance is positive for every maturity T.[1] Sensitivity is computed thanks to adjoint automatic differentiation from neural network.

The above example shows that the functional neural network is indeed apt to learn from the compression stage a low-dimensional representation of likely observations. The low-dimensional representation gives large reconstruction errors to the surfaces of the testing set atypical with respect to the past observations (the training set in our experiments) and their latent structure.

3.6.2 Outlier Detection and Correction

To confirm our views on outliers, we propose the following sanity check. An observation (first volatility surface in the testing set) is chosen and articially corrupted by doubling the values on four randomly chosen points: see the top-left corner in Figure 3.6.4.

Then we run the optimization (3.2) on this corrupted surface and obtain recalibrated codes. These code produce with the decoder the reconstructed surface (called correction) on the top-right corner. The correction is a smooth surface in which the corrupted values have been overwritten by values close to the original (non corrupted) ones. The bottom-left panel shows that only the corrupted values have been modifed significantly by the correction stage. The bottom-right figure indicates that the corrected surface is very close to the original one. The RMSE between the corrupted and the corrected surface is 0.0446 whereas the one between the correction and the original surface is 0.0151.

Note that the calendar arbitrage condition is still respected (see figure 3.6.5)

[1]Regarding butterfly arbitrages, Durrleman's condition on the density (involving sensitivity with respect to forward log-moneyness, cf. Roper (2010)) can unfortunately not be checked for lack of data regarding dividends and discounting.

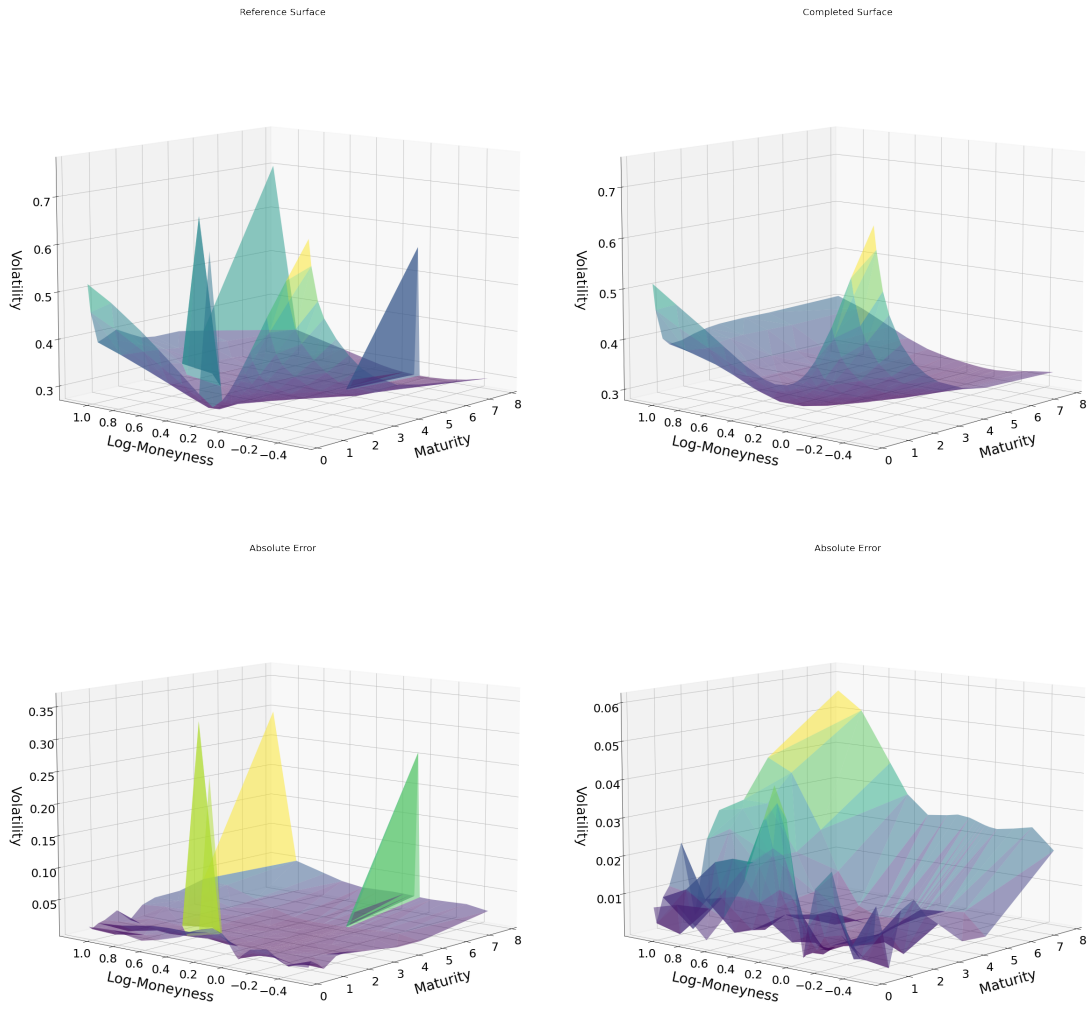


Figure 3.6.4: Outlier correction : Corrupted surface (Top-left), Corrected surface (Top-right), absolute error between corruption and correction (bottom-left), absolute error between correction and original surface before corruption (bottom-right)

for the correction, which exhibits a positive sensitivity of the implied total variance with respect to the maturity of the option.

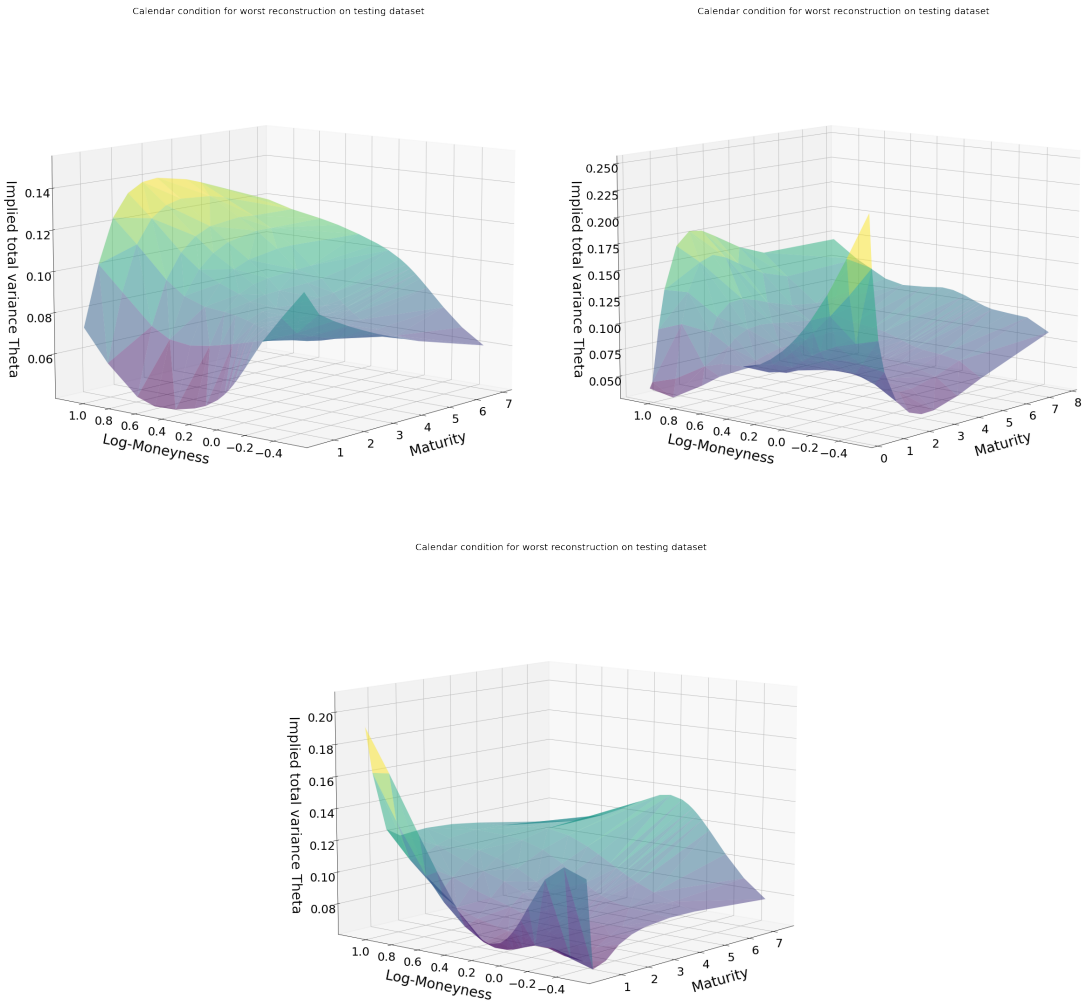


Figure 3.6.5: Implied total variance theta for worst reconstruction on top-left, outlier correction on top-right and worst completion at the bottom.

This experiment confirms that a high reconstruction error is a good indicator of an outlier. The calibrated latent structure of the functional network smoothes the corresponding surface by identifying and correcting its anomalous points.

3.6.3 Completion

We now want to leverage on the calibrated low-dimensional latent structure of the functional network to recover a complete surface from partial information. Our hope is that this procedure will generate likely surfaces while approaching the available values (including on moving grids).

For each observation (surface) in the testing set, we select 40 points among the 255 points and remove all the others. Then we calibrate the latent variables by solving numerically the problem (3.2) with loss corresponding to these 40 points.

In order to benchmark the functional approach and assess the contribution of the historical data to the performance of the method, we report average completion errors [2] on the testing set for standard interpolation procedures (within each given surface, without exploitation of the information provided by the others):

1. Linear interpolation: given a triangulation of the 2D maturity and log-moneyness space base on the locations of the 40 available points, the interpolated value is taken as the barycenter on each triangle;

2. Spline interpolation: uses in each triangle as above a piecewise cubic interpolating Bezier polynomial (see Alfeld (1984) and the scipy documentation of the CloughTocher2DInterpolator method);

3. Gaussian process regression and squared exponential kernel: denoting by X the observed locations (maturity and log-moneyness), by Y the observed lognormal volatilities at locations X, by $X^\star$ the locations without values and by $Y^\star$ the unknown (looked for) implied volatilities, a Gaussian process regression assumes a Gaussian distribution

$$(Y,Y^\star)\sim\mathcal{N}(0,\begin{pmatrix}K(X,X) & K(X,X^\star)\\ K(X^\star,X) & K(X^\star,X^\star)\end{pmatrix}) \text{ with } K(X,X^\star)_{ij}=\sigma\exp\left(-\frac{\|x_i-x_j\|^2}{l^2}\right), \quad (3.7)$$

   where σ and l are two hyperparameters calibrated by log-likelihood to the available values. In (3.7),

$$\|x_i - x_j\|^2 = (T_i - T_j)^2 + (\ln(m_i) - \ln(m_j))^2,$$

   where T denotes a maturity and $\ln(m)$ a log-moneyness;

4. Gaussian process regression with flat extrapolation; similar to 3, except that the Gaussian process predictor is only used for interpolation purposes; extrapolation whenever required is performed by the nearest neighbour method.

[2] Gap between the original surface and the completed one.

Again, a major difference between our functional (or neural net more generally) approach and these interpolation benchmarks is that, in order to interpolate a given surface, the neural network takes into account the information contained in all the surfaces of the data set, which is used as training set at the compression stage. In contrast, the above interpolation benchmarks only use the information provided by the available points of the currently interpolated surface, without consideration of the other surfaces in the data set. In particular, by Gaussian process regression in 3. and 4., we just mean interpolation within a given surface, using the available points in this surface as training set (unrelated to the potential use of Gaussian processes as an alternative to neural networks in our compression/completion approaches, which would be unrealistic as discussed in Subsection 3.2.1).

Accordingly, the functional approach exhibits significantly smaller completion errors. In Table 3.6.2, we reported these errors for two different choices of the 40 visible points :

- Less correlated points, i.e. locations for which the implied volatilities are the less correlated;
- Uniformly spread points, i.e.a random selection of at least 2 points per maturity. The lowest maturity can be assigned 3 visible points in order to reach a total number of 40 points.

As the loss in (3.2) is now computed on much fewer points (partial information in this sense), the compression errors of the functional approach are obviously higher than the reconstruction errors from Table 3.6.1. Smaller error are reported in the second case above because less correlated points are rather located on short maturities, so that, in the first case little information, is available for the long maturities.

	Functional	Functional with Forward	Linear interpolation	Spline interpolation	Gaussian process no extrapolation	Gaussian process flat extrapolation
Less correlated points	0.0262	0.0265	0.0632	0.0462	0.0555	0.0459
Uniformly spread points	0.0076	0.0091	0.0211	0.0168	0.0201	0.0208

Table 3.6.2: RMSEs for completed implied volatilities.

All the completion results reported hereafter correspond to the case of uniformly spread visible points.

The completion method provided by the functional approach is also robust: even the worst completion does not produce an outlier, i.e.

- the completed surface is smooth,
- the completed surface has a shape similar to the one of the original surface (the pointwise errors between the original and the completed surfaces are uniformly distributed),

- the implied total variance sensitivity with respect to the maturity is still positive (see Figure 3.6.5), inducing no calendar arbitrage opportunity,
- tails are consistent with the original points (see Figure 3.6.7) and not irregular.

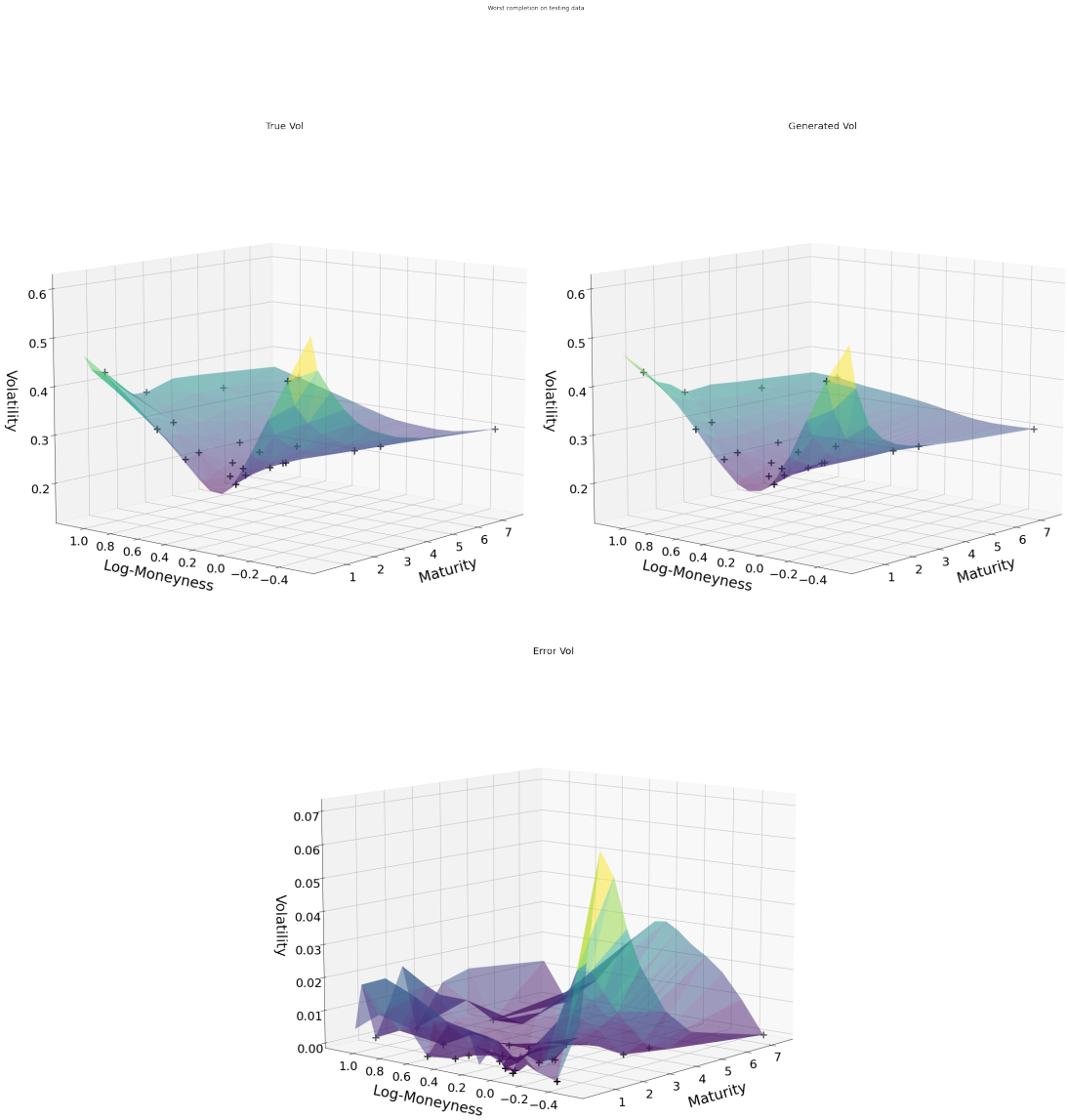


Figure 3.6.6: Original surface vs. completed surface yielding the worst RMSE. Black crosses mark visible points.

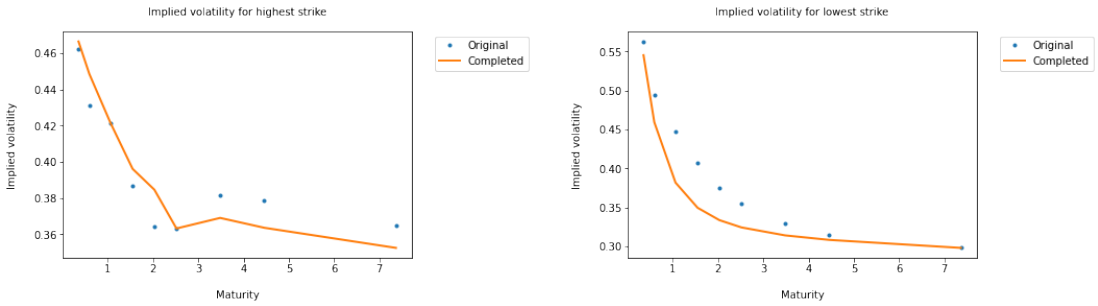


Figure 3.6.7: Tails of completed surface vs. original implied volatilities.

Such robustness is not provided by the interpolation benchmarks. For instance, in the case of the worst completion with the spline interpolation, the completed surface (top-right corner of Figure 3.6.8) is irregular in the tails.

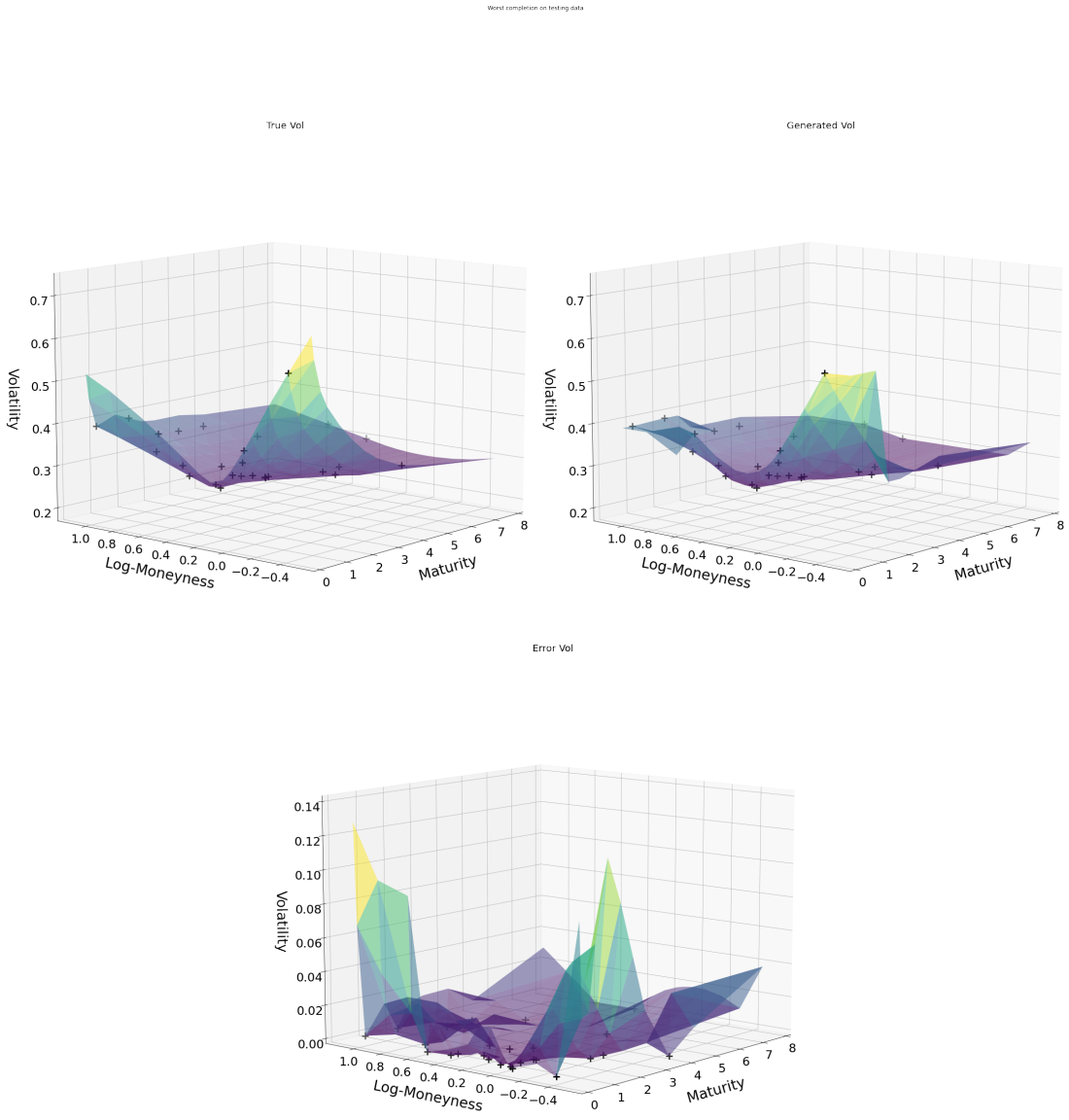


Figure 3.6.8: Original surface vs. completed surface yielding the worst RMSE with spline interpolation. Black crosses mark visible points.

3.7 At-the-Money Swaption Surfaces

The previous section was showing a case where the functional approach outperforms elementary interpolation benchmarks in an situation (in fact, the most common in the context of financial nowcasting applications) involving a moving grid.

We now consider an application where the grid is constant (after a preprocessing by our data provider) so that PCA or more classical autoencoder approaches are also available. The results show that the functional approach then performs as well as these classical benchmarks (which, however, would not be available on the original data with variable time-to-maturity).

A swaption is a financial contract allowing a client to enter into an interest rate swap with some strike K at some future expiry date U, for some tenor length T. A large body of literature deals with the swaption implied volatility as a function of the strike parameter.

By contrast, very few works are dealing with the swaption implied volatility as a function of the expiry and tenor parameters (see Figure 3.7.1). One exception is

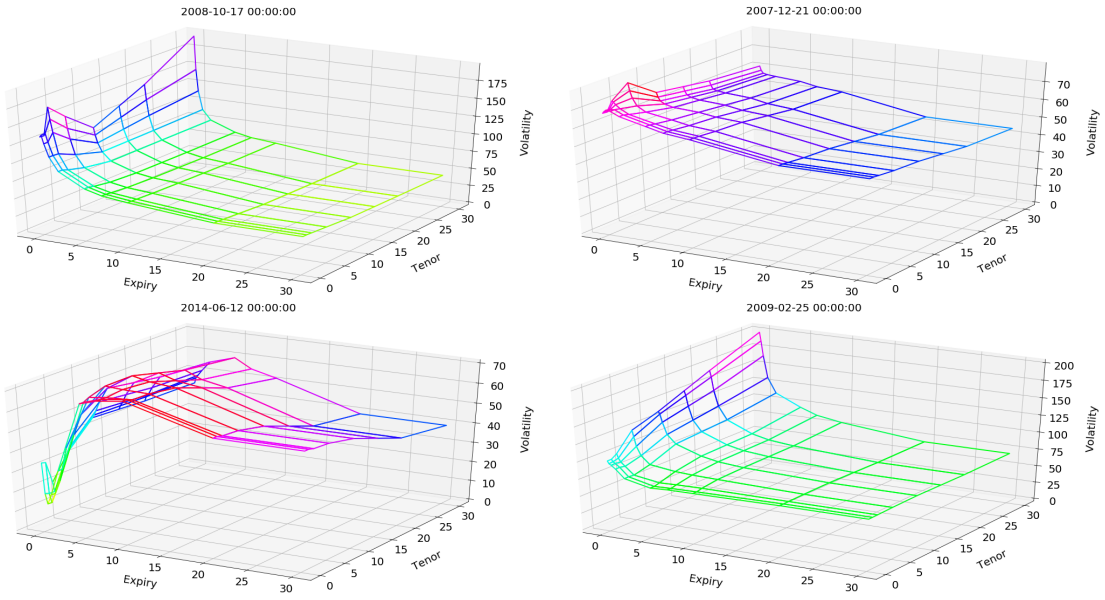


Figure 3.7.1: Different patterns of at-the-money swaption volatility surface.

Trolle and Schwartz (2010), who, based on a time series of swaption cubes, investigate how the conditional moments of the underlying swap rate distributions vary with expiry, tenor, and calendar time. One possible reason for this relative lack of literature may be that swaption arbitrage pricing relationships are mainly known along the strike direction. Across expiries and tenors, one only has "statistical arbitrage" relations, reflecting the overlap between the cash flow streams of the

underlying swaps.

In the following case study, we focus on at-the-money (ATM, which are also the most liquid) swaption implied volatilities as a function of U and T. The approach is model free in the sense that we do not formulate or use any hypothesis on the underlying forward swap rate processes.

Our study is conducted on a daily database of monocurrency (euro) ATM swaption normal[3] implied volatilities, covering 2400 business days corresponding to the period from 2007 to 2017. The training calibration and validation set Ω covers the 2007 to 2014 sub-period (1900 first observation days of the data set), whereas the test set Ω' ranges from 2015 to 2017 (500 subsequent ones). The data have been preprocessed by our provider so that all the ATM implied volatility surfaces are defined on a common rectangular grid of eighty (U, T) nodes, without missing implied volatility values at any day or node, corresponding to the ten expiries (with M for month and Y for year)

$$U \in (1M, 3M, 6M, 1Y, 2Y, 5Y, 7Y, 10Y, 20Y, 30Y)$$

and the eight tenors

$$T \in (3M, 1Y, 2Y, 5Y, 10Y, 15Y, 20Y, 30Y).$$

For testing our completion approach, we mask 90% of the points in each surface of the test set Ω', only keeping the volatility points corresponding to the grid nodes (U, T) in

$$\begin{aligned} &(1M, 3M), (1M, 10Y), (1M, 30Y), (6M, 2Y), \\ &(6M, 15Y), (5Y, 1Y), (5Y, 20Y), (10Y, 5Y). \end{aligned} \tag{3.8}$$

Such specification is in line with the reality of a market where the shortest expiries are the most liquidly traded ones (as well as the most volatile). Hence, our completion exercise corresponds to the intraday situation of a swaption trader facing mostly short expiry ATM implied volatility data, and left with the task of guessing the "most likely values" of the remaining implied volatilities.

3.7.1 Network Architectures

The corresponding architecture of the functional approach is then similar to the one used for equity derivatives in Section 3.5.1, except that the expiry U and tenor T are used as the localization inputs, and only 8 latent variables are used (instead of 15 previously): see Figure 3.7.2. Moreover, one can also incorporate

[3] rather than Black–Scholes, because of the negative rates environment.

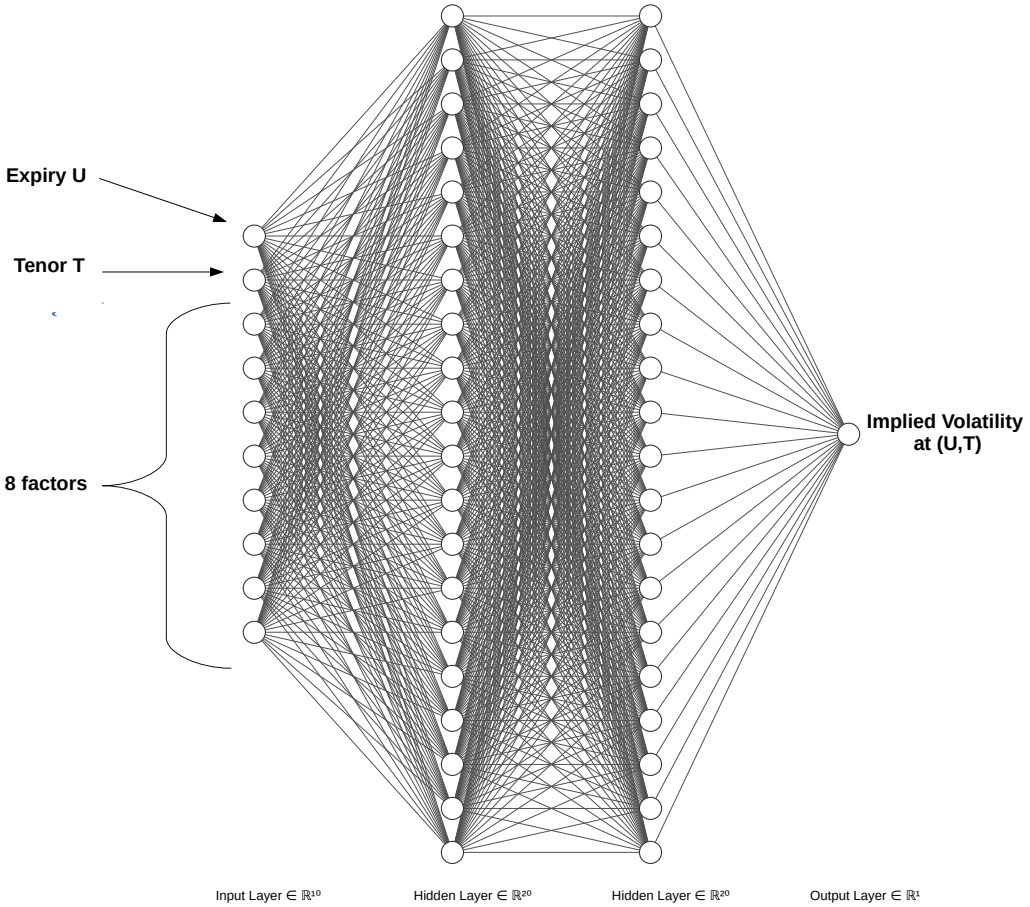


Figure 3.7.2: Network of the functional approach used in the swaption case study (style FCNN of the NN-SVG software, cf. Figure 3.5.1).

the forward swap rates as exogenous variables. These are the underlyings of the swaptions and they are structured similarly to the ATM implied volatilities of the latter, located by an expiry and a tenor. For taking them into account, it suffices to add to the network of Figure 3.7.2 an additional feature (input unit) containing the level of the forward swap rate with expiry U and tenor T. Hence, the units for the expiry U and the tenor T indicate the common location of the corresponding ATM volatilities and forward swap rates.

The convolutional autoencoders use feed-forward neural networks for the encoder and the decoder, with four hidden layers each: one dense layer is applied on top of three convolutional layers for the encoder and, symmetrically, three deconvolutional layers are built on top of one dense layer. The data set is reshaped as a $(10, 8)$ tensor per day. The convolution layers are built with the respective kernels (used for specifying the localization of the weights) $(5, 4)$, $(4, 3)$, and $(3, 3)$. Each

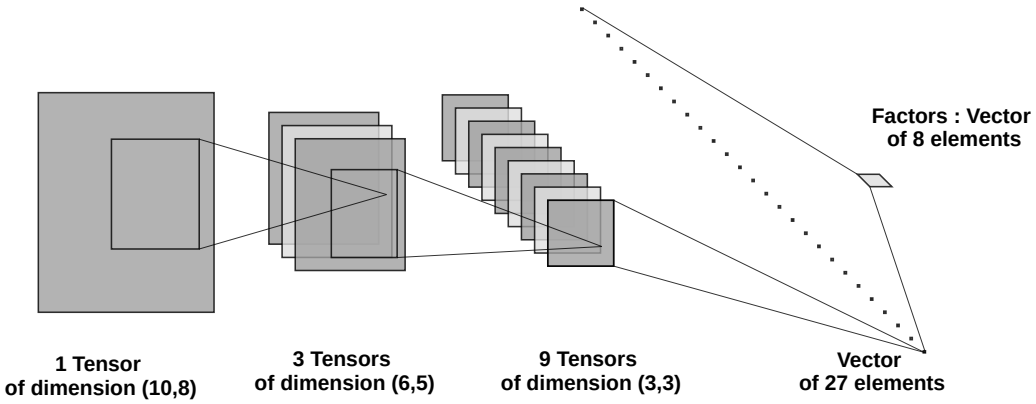


Figure 3.7.3: Architecture of the convolutional encoder used in our ATM swaption case study. Graph produced using the style LeNet of the NN-SVG software: Each of the four layers is represented by a triangle; The inputs of each of the three convolutional layers are displayed as collections of tensors; The ones of the last, dense layer are represented as a series of dots.

convolution layer produces 3 channels (see Figure 3.7.3) and, symmetrically, each deconvolution layer has in input 3 times more channels than in output. Padding is set as VALID in order to reduce the size of the hidden units after each convolution layer. As output of the three convolution layers, we have a hidden layer of 27 units, corresponding to 27 channels of size $(1, 1)$. A softplus (regularized ReLU) activation function is chosen after each convolution layer. This results in sparsity of the calibrated network (the compression stage sets very negative biases on the intermediate units that the neural network wants to ignore, cf. Bengio (2012)),

as well as positivity and regularity of the ensuing implied volatility surface. The dense layers between the factors and the (de)convolution layers are linear. Hence, the convolution layers can be seen as a kernel that linearly separates the features.

Following a divide-and-conquer, sequential training strategy, we train the convolutional layers by pairs, from the most outer to the most inner ones, i.e. the layers surrounding the latent variables (greedy layer-wise pre-training as per Hinton, Osindero, and Teh (2006) and Bengio, Lamblin, Popovici, and Larochelle (2007)). A final optimization fine-tunes the weights of all the layers together. This also allows exploiting any hierarchical structure of the data (cf. Masci, Meier, Cireşan, and Schmidhuber (2011)): The outer layers detect the greatest patterns, while inner layers detect the finest ones.

In the case of the fully connected networks that are used in the linear projection and in the functional approaches, we use the Glorot and Bengio (2010) initialization rule for the weights, with a centered normal distribution of standard deviation equal to $\sqrt{\frac{4}{n_{inputs}+n_{outputs}}}$. In the case of the convolutional layers we use a truncated normal distribution with 0.1 standard deviation. All biases are initialized to zero.

Each iteration leads to the computation of the loss gradient on the whole calibration data set. Indeed, given the relatively small size of our data sets, full gradient evaluation is not an issue in practice. Moreover, mini-batch would require that each batch sample has approximately the same distribution, which is notoriously violated in the case of (non-stationary) financial time series.

Penalization is used at the compression stage for regularizing the calibrated parameters. More precisely, ridge regularization is used for the kernel weights of the fully-connected layers of the convolutional and of the functional approaches, with a penalization coefficient of 0.1 intended to balance the reconstruction loss and the penalization term at the minimum.

3.7.2 Numerical Results

Table 3.7.1 is a report on the errors of all our approaches (cf. Section 3.4.2). It is based on the absolute daily RMSEs (cf. (3.5) and (3.6)).

The last row of Table 3.7.1 displays the corresponding training times for all but the standard PCA approach, which involves no training and is in fact much faster than all the others (as it essentially reduces to the inversion of an $m \times m$ matrix, with $m = 80$). The dates in brackets in the tables identify the observations corresponding to the worst errors.

At the completion stage, we take as initial factor values the volatility encoding of the previous day. Figure 3.7.4 shows the stability through calendar time of the codes obtained by the linear projection approach.

As shown by Figure 3.7.5 in the case of the linear projection approach (but this

	Standard PCA	Linear projection	Convolutional autoencoder	Functional approach	Functional approach with forward rate
Average compression error on Ω	1.23	1.58	1.97	1.85	2.29
Average compression error on Ω'	3.71	3.54	6.19	3.77	3.02
Worst compression error on Ω [day] ([day])	4.15 [2008-12-03]	3.98 [2008-12-09]	7.18 [2008-12-08]	8.32 [2008-10-09]	6.93 [2008-10-10]
Worst compression error on Ω' [day] ([day])	5.76 [2016-04-28]	5.18 [2016-04-28]	12.0 [2015-07-07]	6.34 [2015-12-21	5.16 [2015-12-18]
Average completion error on Ω'	6.19	4.07	5.03	6.41	5.19
Worst completion error on Ω' [day] ([day])	12.6 [2015-06-30]	6.50 [2015-07-10]	9.89 [2015-07-10]	12.8 [2015-03-09]	9.09 [2016-01-14]
Training time in seconds	$\emptyset$	9	411	1287	276

Table 3.7.1: RMSEs in the sense of (3.5) and (3.6)

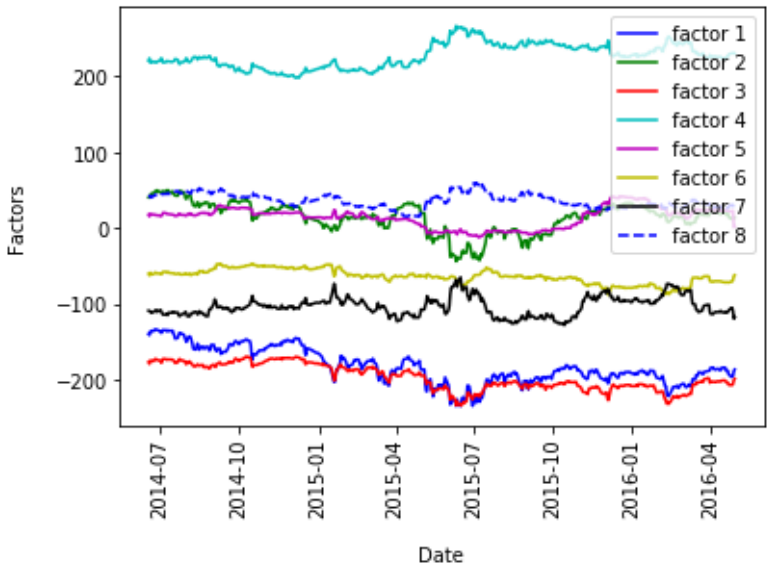


Figure 3.7.4: Time series of the factors obtained by encoding of the training observations under the linear projection approach.

is also true of the nonlinear approaches), the dominant errors are concentrated on

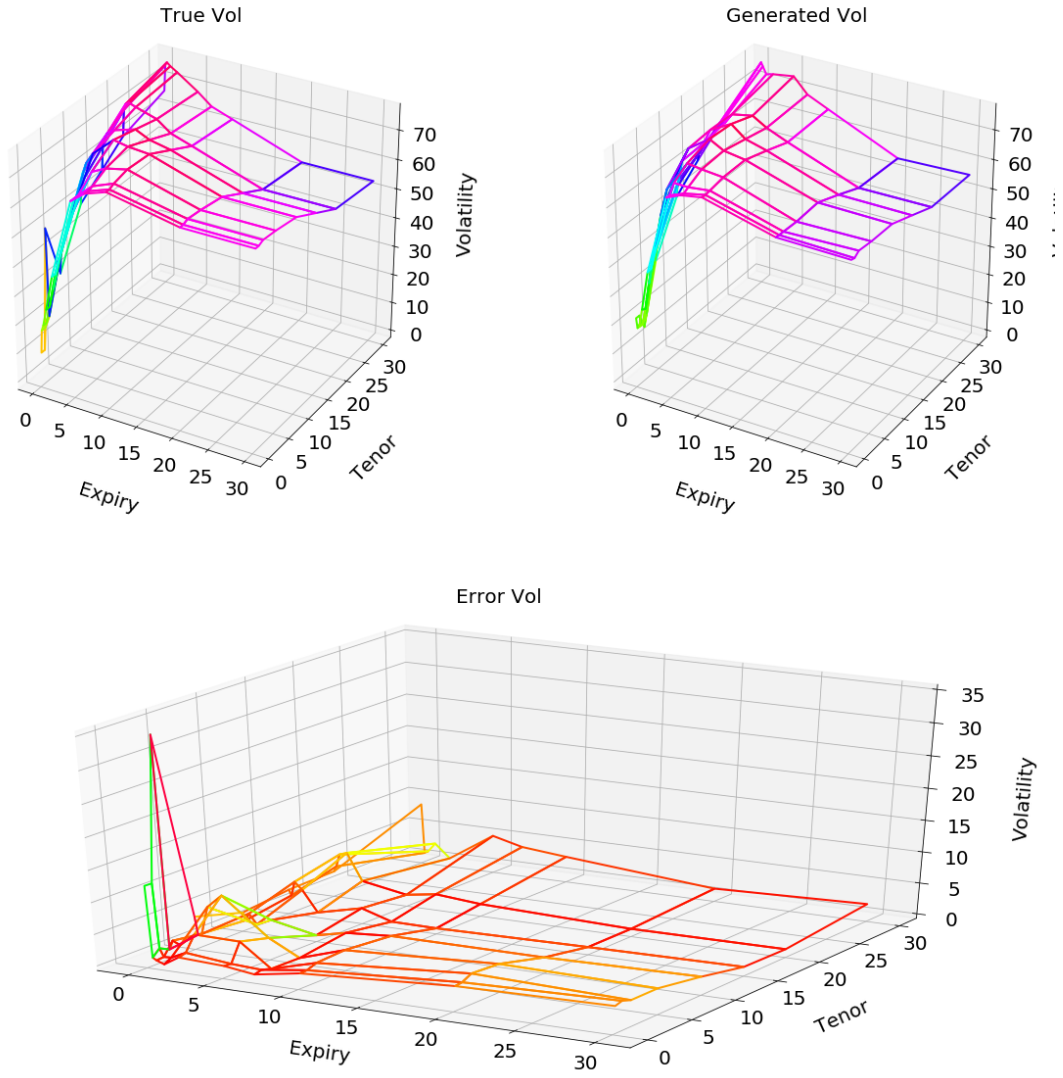


Figure 3.7.5: Linear projection approach: *(Top left)* Original (full) tensor; *(Top right)* Tensor $D_{\delta^\star}(c^\star)$ completed based on the 8 points of the latter given by (3.8); *(Bottom)* Pointwise absolute error between the two, for the worst observation in Ω'.

the shortest expiries. This is because the implied volatilities corresponding to these shortest expiries are the more volatile. Hence, their spatial dependence structure is less informative. To recover these points better, one could think of providing extra information through exogenous variables, such as the level of the underlying

forward swap rates. Under the functional approach, this can easily be done in the way explained in Section 3.5.1. However, the last columns in Table shows that this only has a minor positive impact.

The linear approaches are as accurate as the nonlinear ones and the convolutional approach is typically outperformed by at least the linear projection or the functional approach.

Figure 3.7.7 illustrates that the functional approach enables to interpolate smoothly the surface over an arbitrarily fine grid, in this case 10^4 points obtained by the corresponding interpolation of the tensor of Figure 3.7.6.

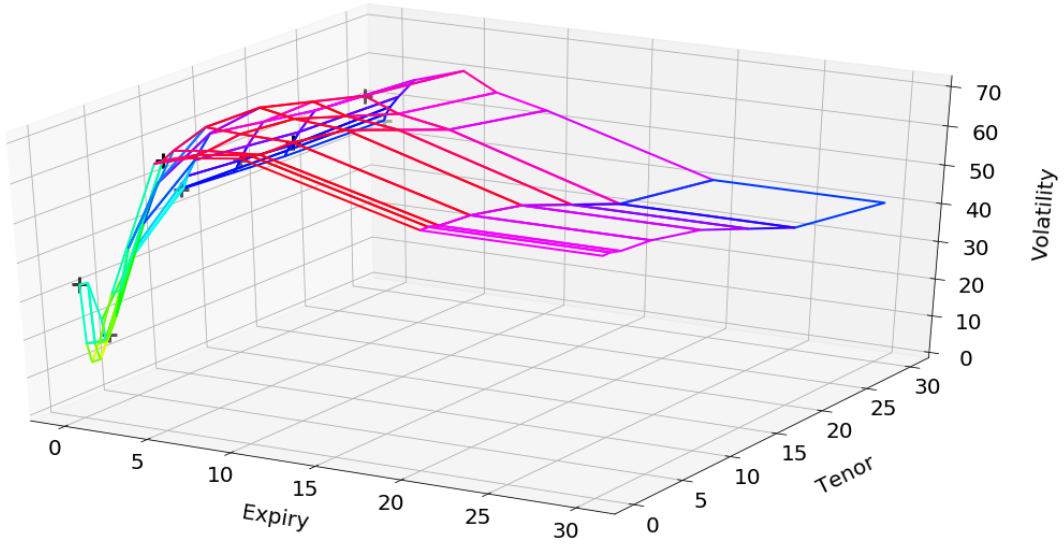


Figure 3.7.6: Complete tensor corresponding to the first observation in Ω'. The black crosses designate the "available points", specified by (3.8), that are used in the completion exercise.

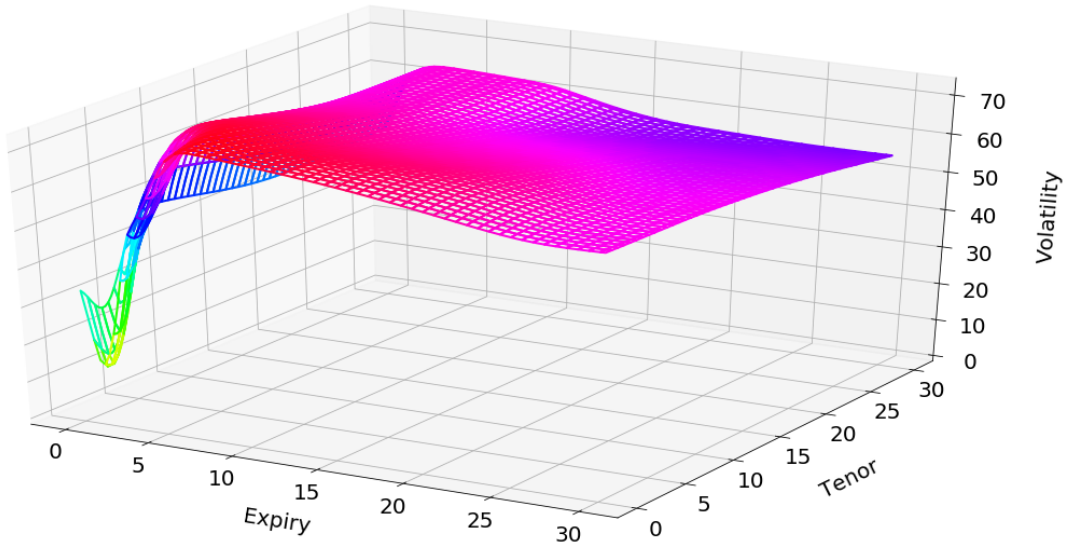


Figure 3.7.7: Surface with 10^4 points obtained by the functional approach applied to the first observation in Ω'.

3.8 Conclusions and Perspectives

We have devised a generic neural network based curve or surface (or more general tensor) compression/completion methodology, for which we propose two concrete specifications: the functional approach, amenable to the treatment of unstructured data with varying grid nodes (as natively the case in most financial nowcasting applications), and a convolutional autoencoder approach, including PCA or PCA-like projections as linear special cases, applicable in the special case of a constant grid (natively or possibly after some preprocessing). The compression stage also allows for outlier detection and correction by generating surfaces or curves in line with training samples.

The analysis of the corresponding reconstruction errors suggests that linear methods are sufficient to compress structured tensors, corresponding to a constant grid of nodes, into few factors coefficients. The completion stage allows recovering with success about 90% values of the data, starting from about 10% of known values. But the functional approach is the only one that is able to directly deal (without preprocessing) with the most common situation of unstructured tensors. The only alternative is then naive interpolation benchmarks that do not exploit the data set, and which the functional approach is shown to outperform in our equity derivative case study.

All approaches suffer from non-stationarities occurring during extreme events or change of market regimes. This can be seen as an advantage with respect to

anomaly detection. For other purposes, it would plead in favor of further modeling of the factor dynamics, whether this relies on times series machine learning or Markov chain Monte Carlo (filtering) techniques. More generally, it would be interesting to extend this study in several directions, such as the introduction of backtesting hedging criteria (cf. Garcia and Gençay (2000)), scenario simulation in a context of variational networks (see Tschannen, Bachem, and Lucic (2018)), application of the method to the whole swaption volatility cube, strike dimension included (cf. Trolle and Schwartz (2010)), or specification of dynamics on the factors (for instance by Kalman filters).

Chapter 4

XVA compression

Since the 2008–09 financial crisis, banks have introduced a family of X-valuation adjustments (XVAs) to quantify the cost of counterparty risk and of its capital and funding implications. XVAs represent a switch of paradigm in derivative management, from hedging to balance sheet optimization. They reflect market inefficiencies that should be compressed as much as possible. In this work we present a genetic algorithm applied to the compression of credit valuation adjustment (CVA), the expected cost of client defaults to a bank. The design of the algorithm is fine-tuned to the hybrid structure, both discrete and continuous parameter, of the corresponding high-dimensional and nonconvex optimization problem. To make intensive trade incremental XVA computations practical in real-time as required for XVA compression purposes, we propose an approach that circumvents portfolio revaluation at the cost of disk memory, storing the portfolio exposure of the night so that the exposure of the portfolio augmented by a new deal can be obtained at the cost of computing the exposure of the new deal only. This is illustrated by a CVA compression case study on real swap portfolios.

4.1 Introduction

XVAs, where VA stands for valuation adjustment and X is a catch-all letter to be replaced by C for credit, F for funding, M for margin, and K for capital, have been implemented by banks in reaction to the regulatory changes aroused by 2008 financial turmoils. They monetize counterparty risk and its funding and capital consequences by add-ons to derivative entry prices sourced from clients. According to the cost-of-capital XVA approach of Crépey, Hoskinson, and Saadeddine (2021), accounting for the impossibility for a bank to replicate the jump-to-default related cash flows, the final, all-inclusive XVA formula reads

$$\mathrm{CVA} + \mathrm{FVA} + \mathrm{MVA} + \mathrm{KVA}, \tag{4.1}$$

to be sourced by the bank from clients on an incremental run-off basis at every new deal.

As stated by the Basel Committee on Banking Supervision (2015), major counterparty credit losses on OTC derivative portfolios in 2008 arose from XVA accounting losses rather than from actual client defaults. In particular, a bank incurs a CVA loss when the market perceives a deterioration of the credit risk of a client. This has motivated the creation of XVA desks for dealing with these risks.

In this chapter, we deal with CVA compression, i.e. the minimization of the CVA of a client portfolio by the introduction of an incremental trade, subject to the constraint of not altering too much the market risk of the portfolio. In the financial derivative industry, the term compression term is generally applied in the context of "trade compression", i.e. the reduction of the gross notional of positions in the market. Trade compression aims notably at reducing certain capital requirements, the number of transactions, and their amount (see section 5.3 of Gregory (2015)). As reflected by the proliferation of related industry presentations[1], this kind of balance sheet optimization is very active in top tier banks at the moment.

XVAs reflect market inefficiencies that should be compressed as much as possible. Here we focus on CVA compression for concreteness and simplicity, but the developed XVA compression methodology is generic. It could and should be applied to further XVA metrics, as soon as these are available with sufficient speed, for computation at the portfolio level, and accuracy, for numerical significance of the results at the incremental trade level: see Section 5 in Albanese, Chataigner, and Crépey (2019), which emphasizes the XVA compression perspective on the pricing and risk management of financial derivatives in the post-2008–09 global financial crisis era, and cf. Kondratyev and Giorgidze (2017), who use a genetic algorithm for determining an optimal trade-off between MVA compression and transaction costs.

The complexity of XVA compression problems stems, in particular, from the hybrid nature of the state space of the corresponding optimization problems. Indeed, a new trade (financial derivative) is described by a combination of continuous and discrete parameters. This rules out the use of standard convex optimization schemes for such problems. Instead, we are lead to the use of metaheuristic algorithms: In this chapter, we show how a genetic algorithm with penalization can efficiently find a CVA offsetting trade, while limiting the impact of the trade on the market exposure profile. The latter is necessary for staying in line with the separation of mandates between the XVA desks, in charge of managing counterparty risk, and the other, dubbed "clean", trading desks of the bank, in charge of hedging the market risk of the bank positions.

[1] cf. e.g. David Bachelier, panel discussion Capital & margin optimisation, Quantminds International 2018 conference, Lisbon, 16 May 2018.

The other XVA compression challenge is execution time, with intensive valuation of the involved XVA metrics as a bottleneck. The XVA metrics are primarily defined at the portfolio level: Time-0 XVAs can be formulated as expectations of nonlinear functionals of the bank derivative portfolio exposure, i.e. "clean" valuation (or "mark-to-market" MtM ignoring counterparty risk) of the bank portfolio, assessed at randomly sampled times and scenarios. Each new deal gives rise to XVA add-ons computed as the corresponding trade incremental XVA amounts, i.e. the differences between the XVAs of the portfolios including and excluding the new deal. To make intensive trade incremental XVA computations practical in real-time as required for XVA compression purposes, our proposed *MtM store-and-reuse approach* circumvents clean revaluation at the cost of disk memory, storing the portfolio exposure of the night so that the exposure of the portfolio augmented by a new deal can be obtained at the cost of computing the exposure of the new deal only.

4.1.1 Outline and Contributions

The chapter is outlined as follows. Section 4.2 formulates the penalized CVA compression problem and introduces the related genetic optimization algorithm. Section 4.3 is about two key acceleration techniques in this regard. Section 4.4 presents a numerical case study on real swap portfolios. Section 4.5 concludes.

The main contributions of the chapter are the design of a parallelized genetic algorithm for the CVA compression task, the MtM store-and-reuse acceleration technique for trade incremental XVA computations, and the numerical CVA compression case study on real swap portfolios.

More broadly, this chapter enriches the literature on the use of genetic (also called evolutionary) optimization algorithms in finance. Hamida and Cont (2005) applied evolutionary algorithms to investigate a set of co-calibrated model parameterizations in order to assess the associated model risk. Kroha and Friedrich (2014) compared different genetic algorithms for automatic trading. Jin, Yang, and Yuan (2019) applied evolutionary algorithms to optimal investment and consumption stochastic control problems. For wider reviews of genetic algorithms in finance, we refer the readers to Drake and Marks (2002) and Chen (2012).

We refer the reader to the end of the chapter for a list of the main abbreviations.

4.2 CVA Compression Modeling

4.2.1 Credit Valuation Adjustment

We consider a complete stochastic basis $(\Omega, \mathbb{F}, \mathbb{P})$, for a reference market filtration (ignoring the default of the bank itself) $\mathbb{F} = (\mathfrak{F}_t)_{t\in\mathbb{R}_+}$, satisfying the usual conditions, and a risk-neutral pricing measure $\mathbb{P}$, calibrated to market quotes of fully collateralized transactions. All the processes of interest are $\mathbb{F}$ adapted and all the random times of interest are $\mathbb{F}$ stopping times. This holds at least after so-called reduction of all the data to $\mathbb{F}$, starting from a larger filtration $\mathbb{G}$ including the default of the bank itself as a stopping time, assuming immersion from $\mathbb{F}$ into $\mathbb{G}$ for simplicity (see Crépey, Hoskinson, and Saadeddine (2021) for the detail). The $\mathbb{P}$ expectation and $(\mathfrak{F}_t, \mathbb{P})$ conditional expectation are denoted by $\mathbb{E}$ and $\mathbb{E}_t$.

In developed markets, the overnight indexed swap (OIS) rate is together the reference remuneration rate for posted collateral and the best market proxy for a risk-free rate. We denote by $r = (r_t)_{t\in\mathbb{R}_+}$ an $\mathbb{F}$ progressive OIS rate process and we write $\beta = e^{-\int_0^{\cdot} r_s ds}$ for the corresponding risk-neutral discount factor.

By clean valuation or mark-to-market of a contract (or portfolio), we mean the (trade additive) risk-neutral conditional expectation of its OIS discounted future promised cash flows, ignoring counterparty risk and its capital and funding implications.

We consider a bank engaged into bilateral trading with a single corporate counterparty (client). with default time and recovery rate τ_c and R_c. This setup, which is chosen for simplicity, is consistent with a common situation where credit risk budget is assigned at each counterparty level within the bank. We denote by MtM the corresponding mark-to-market process of the client portfolio to the bank.

The (time 0) CVA of the bank is its expected discounted loss in case of client default, i.e.

$$\mathrm{CVA} = \mathbb{E}\big[\mathbb{1}_{\{\tau_c\le T\}}\beta_t^{-1}\beta_{\tau_c}(1-R_c)\mathrm{MtM}_{\tau_c}^+\big]. \tag{4.2}$$

Assuming deterministic interest rates, this can be rewritten as

$$\mathrm{CVA} = (1-R_c)\int_0^T \beta_t \mathrm{EPE}(t)\mathbb{P}(\tau_c\in dt), \tag{4.3}$$

where the expected positive exposure (EPE) is defined as

$$\mathrm{EPE}(t) = \mathbb{E}(\mathrm{MtM}_s^+|s=\tau_c)_{|\tau_c=t}. \tag{4.4}$$

The formula (4.3) is popular with practitioners because it allows obtaining the CVA as the integral of the EPE against the client CDS curve. But it is only really practical in simplistic models where the market and credit sides of the problem are

independent, so that $\mathrm{EPE}(t) = \mathbb{E}(\mathrm{MtM}_t^+)$. However, a key CVA modeling issue is wrong-way risk, i.e. the risk of adverse dependence between market and credit (see Pykhtin (2012), Hull and White (2012), Li and Mercurio (2015), Taarit (2018), Crépey and Song (2016, Crépey and Song (2017), Brigo and Vrins (2018),Glasserman and Yang (2018)).

Assuming the client default time endowed with an intensity γ^c, a more flexible formula is

$$\mathrm{CVA} = (1 - R_c)\mathbb{E}\int_0^T \beta_s e^{-\int_0^s \gamma_u^c du}\gamma_s^c \mathrm{MtM}_s^+ ds. \tag{4.5}$$

Under a credit support agreement (CSA), MtM should be replaced by $(\mathrm{MtM} - \mathcal{C})$ in all equations above, where $\mathcal{C}$ is the collateral posted by the counterparty. Obviously, collateral can mitigate the EPE and the CVA considerably. In the data of our case study there is no CSA, i.e. $\mathcal{C} = 0$.

Non-linearity of MtM^+ with respect to the portfolio payoff components imposes CVA calculations at the counterparty portfolio (netting set) level.

Similar approaches apply to FVA computations, with analogous comments, whereas the MVA can be computed based on quantile regression for the embedded dynamic initial margin calculations (see Crépey, Hoskinson, and Saadeddine (2021)). In any case, the numerical bottleneck of XVA computations lies in intensive MtM calculations.

4.2.2 Fitness Criterion

By the augmented, respectively initial, portfolio, we mean the portfolio of the bank inclusive, respectively exclusive, of a newly considered deal with the client. The aim of an XVA compression problem is to find a new trade that minimizes the corresponding XVA metric of the augmented portfolio. This is equivalent to minimize the incremental CVA, which we denote by

$$\Delta\mathrm{CVA} = (1 - R_c)\mathbb{E}\int_0^T \beta_s e^{-\int_0^s \gamma_u^c du}\gamma_s^c (\mathrm{MtM}_s^{augm})^+ ds - (1 - R_c)\mathbb{E}\int_0^T \beta_s e^{-\int_0^s \gamma_u^c du}\gamma_s^c (\mathrm{MtM}_s^{init})^+ ds \tag{4.6}$$

$$= (1 - R_c)\mathbb{E}\int_0^T \beta_s e^{-\int_0^s \gamma_u^c du}\gamma_s^c \big((\mathrm{MtM}_s^{augm})^+ - (\mathrm{MtM}_s^{init})^+\big) ds, \tag{4.7}$$

where the indices *init* and *augm* refer to the initial portfolio and augmented portfolio. We emphasize that trade incremental CVA computations require two portfolio-wide calculations: one without the new trade and another one including it.

Minimizing an XVA metric is most easily obtained through a significant deformation of the portfolio exposure process (especially in the context of this work of a portfolio with a single counterparty). But an XVA compression procedure should not affect too much the market risk of the portfolio, because market risk is the

mandate of the clean desks of the bank, who, in particular, are subject to trading limits.

This motivates the addition of a penalization to the incremental XVA criterion. In our case study, the incremental deal will consist of an interest rate swap. As such product is mostly sensitive to interest rate moves, a natural penalization is then in terms of its DV01 (dollar value of an 01), i.e. the variation of its mark-to-market (at time 0) under a parallel shift of the yield curve by one basis point ($= 10^{-4}$).

More precisely, an interest rate swap exchanges one leg indexed on a floating interest rate against one leg paying a fixed interest rate, called swap rate. The swap is said to be payer (resp. receiver) for the party that pays (resp. receives) the floating payments. A monocurrency swap exchanges both legs in the same currency. It is mainly sensitive to the fluctuations of the corresponding floating interest rate term structure. DV01 measures the associated risk as the difference between the prices of the swap under the baseline (actual market data observed in the real market) and for a bumped yield curve defined as the concatenation of the money market rates, forward rates, and swap rates, on the relevant (successive) time segments. Bumping the yield curve typically means adding 10^{-4} to each tenor of this curve and updating the other reference curves (zero coupon rates, forward rates, ...) accordingly.

Focusing on the CVA metric in this chapter, we obtain the following fitness minimization problem:

$$\underset{x \in \mathcal{A}}{\text{minimize}} \quad f(x) = \Delta\text{CVA}(x) + \alpha|\text{DV01}(x)|, \tag{4.8}$$

where x parameterizes a new deal (swap) to be found in a suitable search space $\mathcal{A}$ (see Sect. 4.4.1), $\Delta\text{CVA}(x)$ is its incremental CVA (cf. (4.6)), $\text{DV01}(x)$ is its DV01, and α is a penalization parameter controlling the trade-off between CVA reduction and market risk profile preservation. By solving (4.8), we aim at identifying a new deal which, if added to the current client portfolio of the bank, would diminish its counterparty risk without impacting too much its market risk. Note that, for scaling reasons (with, in particular, market penalization), we address the XVA compression problem in terms of trade incremental (as opposed to augmented portfolio) XVA numbers.

A new deal is determined by a combination of quantitative (e.g. notional, maturity,...) and qualitative (e.g. currency, long or short direction,...) parameters, so that no gradient or Hessian is available for the fitness function f in (4.8). Moreover, one is interested in exploring a variety of local minima of f, to see different trading opportunities emerge from the optimization procedure. Furthermore, we can guess that some (crucial) parameters need be learned first, such as currency or maturity; other parameters, such as notional, can be refined in a second stage.

All these features lead us to addressing (4.8) by means of a genetic optimization algorithm.

4.2.3 Genetic Optimization Algorithm

Genetic optimization algorithms belong to the class of derivative-free optimizers, which is surveyed and benchmarked numerically in Rios and Sahinidis (2013) (including the CMA-ES and DAKOTA/EA genetic algorithms).

The idea of genetic (or evolutionary) optimization algorithms is to evolve a population of individuals through cycles of modification (mutation) and selection in order to improve the performance of its individuals, as measured by a given fitness function. In addition, so-called crossover is used to enhance the search in parameter space. To the best of our knowledge, evolutionary algorithms were first explicitly introduced in Turing (2009, chapter 7 *Learning Machines*, p.456). See the classical monographs by Holland et al. (1992), Goldberg (1989), and Back (1996). They then experienced the general artificial intelligence disgrace and comeback before and after the 2000s. But they always stayed an active field of research, seen from different perspectives, such as particle filtering, MCMC, or sequential monte carlo methods (see Del Moral and Formulae (2004)). Beyond its financial applications reviewed at the end of Sect. 4.1, genetic optimization has been used in many different fields, such as mechanics Verma and Lakshminiarayanan (2006), calibration of neural networks hyperparameters Young, Rose, Karnowski, Lim, and Patton (2015), or operational research Larranaga, Kuijpers, Murga, Inza, and Dizdarevic (1999).

Genetic optimization offers no theoretically guaranteed rate of convergence, but it is often found the most efficient approach in practice for dealing with hybrid (partly continuous, partly discrete/combinatorial, hence without well defined gradient and Hessian), nonconvex (in the sense of one local minimum, at least, for each set of values of the discrete parameters), and high-dimensional optimization problems such as (4.8).

At each iteration, the fitness $f(x)$ is computed for each individual (also named chromosome) x of an initial population (a set of chromosomes). The values returned by the objective function are used for selecting chromosomes from the population. Among numerous selection methods (see Blickle and Thiele (1995)), we can quote fitness proportionate selection, ranking proportionate selection (in order to avoid the overrepresentation of the chromosomes with the highest fitness values), and tournament selection (selection of the best among randomly drawn chromosomes). The common intention of these selection methods is to sample in priority individuals with the best fitness values. A genetic algorithm is dubbed elitist if the selection operator always keeps the chromosome with the best fitness value. Otherwise (as in our case), there is no guarantee that the best visited

chromosome is contained in the population corresponding to the final iteration.

The mutation stage is intended to maintain some diversity inside the population, in order to avoid the algorithm being trapped by local minima. A mutation randomly changes one gene, i.e. one component (e.g., in our case, the notional of a new swap) of a chromosome.

Selection and mutation play opposite roles: a focus on fitness leads to a quicker convergence toward a local minimum; conversely, a too heavily mutated population results into a slow random research.

In addition, a crossover operator plays the role of a reproduction inside the algorithm. The principle of crossover is to build two children chromosomes from parent chromosomes. A distribution (often the same as the one used for selection) is chosen for picking chromosomes from a population of the previous iteration and for recombining pairs of selected chromosomes. Children share gene values of their parents but a gene value from one parent cannot be inherited by both children. A crossover mask decides for each gene in which parent a child can copy the gene version. One of the most popular crossover masks is single point crossover (see remark 4).

The role of the crossover operator is paradoxical, as crossover can be seen as a combination of mutations, which increase the genetic diversity, while crossover also promotes chromosomes with higher fitnesses. Crossover aims at benefiting of a presupposed proximity of best solutions.

The above operators are applied iteratively until a suitable stopping condition is satisfied. The most basic one is a fixed number of iteration, but customized criteria may also be used to limit further the number of iterations. For instance, the algorithm can be interrupted when the minimum (or sometimes even the maximum) fitness value within the population at the beginning of an iteration is below a predefined threshold.

See Algorithm 4 and Figure 4.2.1 for the algorithm in pseudo-code and skeleton forms, denoting by r_m the mutation rate, i.e. the percentage of individuals in a population affected by a mutation, and by r_c the crossover rate, i.e. the percentage of individuals affected by crossover recombination.

The behavior of a genetic optimization algorithm is essentially determined by the choice of the selection operator, the number of solutions affected by a mutation, and the number of chromosomes affected by a crossover. See Tabassum and Mathew (2014) for a user guide to the main genetic algorithm ingredients and Carvalho, Bittencourt, and Maia (2011) for applications of genetic optimization algorithms to benchmark functions.

Algorithm 4 Pseudo-code of an optimization genetic algorithm.

Data: An initial population $\mathcal{P}_{init}$ of size P and the associated fitness values for each chromosome, a crossover rate r_c, and a mutation rate r_m.

Initialization
while *a stopping condition is not satisfied* **do**
    Save $\lfloor (1 - r_c) * P \rfloor$ chromosomes, chosen by an appropriate selection method from $\mathcal{P}_{init}$, in $\mathcal{P}_{selected}$
    Save $\lfloor r_c * P \rfloor$ chromosomes, chosen by an appropriate selection method from $\mathcal{P}_{init}$, in $\mathcal{P}_{crossover}$
    Recombine, uniformly without replacement, $\lfloor \frac{r_c * P}{2} \rfloor$ pairs from $\mathcal{P}_{crossover}$
    Merge $\mathcal{P}_{crossover}$ and $\mathcal{P}_{selected}$ in $\mathcal{P}_{mutated}$
    Mutate randomly $\lfloor r_m * P \rfloor$ in $\mathcal{P}_{mutated}$
    for *Each chromosome c in* $\mathcal{P}_{mutated}$ **do**
        Compute the fitness value of c

Result: A new population and the associated fitness values.

Remark 4 (Single Point Crossover)**.** *Let* (p_1, p_2) *be a pair of chromosomes chosen as parents and let* (c_1, c_2) *denote the children. We assume that each chromosome has four genes* A, B, C, D*, that* p_1 *has gene versions* $\{A_1, B_1, C_1, D_1\}$ *and* p_2 *has gene versions* $\{A_2, B_2, C_2, D_2\}$*. For a single point crossover, we draw uniformly an integer* i *such the first* i *genes for* c_1 *are inherited from* p_1 *and the remaining genes are transferred from* p_2 *to* c_1*, and symmetrically so for* c_2*. For instance, if we draw* $i = 2$*, then* c_1 *has gene versions* $\{A_1, B_1, C_2, D_2\}$*, and* c_2 *has gene values* $\{A_2, B_2, C_1, D_1\}$*.*

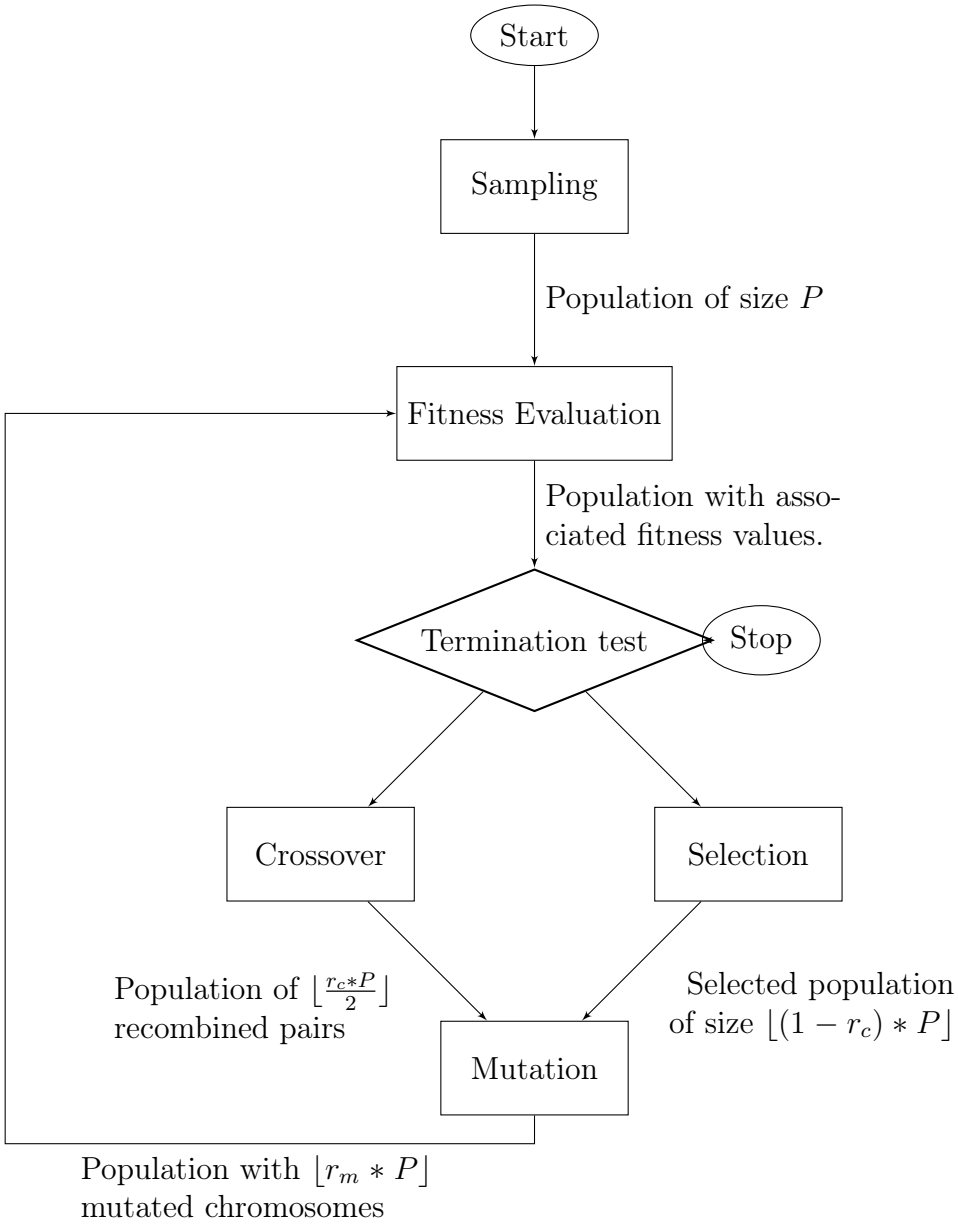


Figure 4.2.1: Skeleton of an optimization genetic algorithm.

4.3 Acceleration Techniques

Without suitable acceleration techniques, the above CVA compression approach is not workable in real time on realistic banking portfolios: on the examples of Section 4.4, a naive (desktop) implementation requires about 20 hours of computations. This becomes even more problematic for hyperparameters tuning (such as

α, crossover rate r_c, etc.). Hyperparameters are generally chosen with grid search, random search (see Bergstra, Bardenet, Bengio, and Kégl (2011)), Bayesian optimization (see Snoek, Larochelle, and Adams (2012)) or even evolutionary algorithms again (see Young, Rose, Karnowski, Lim, and Patton (2015)). In any case, their calibration is greedy in terms of overall genetic algorithm execution.

In this section we deal with the two following acceleration techniques, which may be used simultaneously:

- A MtM store-and-reuse approach for trade incremental XVA computations, speeding up the unitary evaluation of the fitness function;
- A parallelization of the genetic algorithm accelerating the fitness evaluation at the level of the population.

4.3.1 MtM Store-and-Reuse Approach for Trade Incremental XVA Computations

Most of the time in portfolio-wide XVA calculations is spent in clean valuation (i.e. mark-to-market MtM) computations: by comparison, simulation of the risk factors or of the collateral are typically negligible.

Our case study is based on the CVA metric. As observed after (4.6), by lack of trade-additivity of the (portfolio-wide) CVA, trade incremental XVA computations require two portfolio-wide calculations: one without the new trade and another one including it. But it is possible to store the (including MtM) paths simulated for the initial portfolio and reuse them each time we want to compute a new trade incremental XVA. Then, each trade incremental XVA computation only requires the forward simulation of the mark-to-market process of the new deal.

The corresponding *MtM store-and-reuse approach* to trade incremental XVA computations circumvents repeated valuations at the cost of disk memory. It exploits the trade additivity of clean valuation by recording the MtM paths of the initial portfolio on a disk. For every new deal, the augmented portfolio exposure is obtained by adding, along the paths of the risk factors, the mark-to-market of the initial portfolio and of the new deal. This augmented portfolio exposure is then plugged into the XVA engine.

An optimally implemented MtM store-and-reuse approach brings down trade incremental XVA computations to the time of generating the clean price process of the trade itself, instead of the one of the augmented portfolio as a whole. Another advantage of this approach is its compliance with desk segregation: As far as clean valuation is concerned, the XVA desks just use the pricers of the clean desks. Hence, the MtM process plugged into the XVA computations is consistent with the one used for producing the market risk hedging sensitivities.

However, such an approach comes at the costs of memory disk (obviously), but also data slippage as, for consistency, it requires to anchor all the trade incremental XVA computations at the market data and parameters corresponding to the generation of the initial portfolio exposure. In practice, an MtM process at the overall portfolio level can only be generated during night runs, between two market sessions.

Moreover, we have to distinguish between first order (or first generation) XVAs, which are options on the MtM process, and higher order (or second generation) XVAs (see Crépey, Hoskinson, and Saadeddine (2021)), which can be viewed as compound options of order two or more on the MtM process. Second generation XVAs may also involve conditional risk measures, e.g. conditional value-at-risk for the dynamic initial margin calculations that are required for MVA [2] computations, as opposed to conditional expectations only in the case of first generation XVAs.

A Monte Carlo simulation diffuses risk factors X (such as interest rates, credit spreads, etc.) along drivers Z (such as Brownian motions, Poisson processes, etc.), according to a model formulated as a Markovian system of stochastic differential equations, starting from some given initial condition X_0 for all risk factors, suitably discretized in time and space. Modulo calibration, X_0 can be identified with the time 0 market data. We denote by $\widehat{Y}$ a suitable estimate of a process Y at all (outer) nodes of a Monte Carlo XVA engine. In particular, $\widehat{\mathrm{MtM}}$ is the fully discrete counterpart of the MtM process of the initial portfolio, namely the clean value of the portfolio at future exposure dates in a time grid and for different scenario paths.

At first sight, an MtM store-and-reuse approach is unsuitable for second order XVAs, such as the MVA and the KVA (but also the CVA in the case of a CSA where the bank receives so-called initial margin), Indeed, in their case, the principle of swapping computations against storage would require to store not one portfolio exposure $\widehat{\mathrm{MtM}}$, but a whole family of resimulated, future conditional portfolio exposures, (at least, over a certain time horizon), which seems hardly feasible in practice. However, even in the case of second order XVA metrics, an MtM store and reuse approach can be implemented with the help of appropriate regression techniques (at the cost of an additional regression error, see Crépey, Hoskinson, and Saadeddine (2021)).

Formalizing the above discussion, the conditions for a straightforward and satisfactory application of the MtM store-and-reuse approach to a given XVA metric are as follows, referring by indices *init*, *incr*, and *augm* to the initial portfolio, the new deal, and the augmented portfolio:

1. (No nested resimulation of the portfolio exposure required) The formula for

[2] For details regarding the initial margin and the MVA, see Crépey, Hoskinson, and Saadeddine (2021, sections 3.4 and A.4).

the corresponding (portfolio-wide, time-0) XVA metric should be estimatable without nested resimulation, only based on the portfolio exposure rooted at $(0, X_0)$. A priori, additional simulation level makes nonpractical the MtM store-and-reuse idea of swapping execution time against storage;

2. (Common random numbers) $\widehat{\mathrm{MtM}}^{incr}$ should be based on the same paths of the drivers as $\widehat{\mathrm{MtM}}^{init}$. Otherwise, numerical noise (or variance) would arise during $\widehat{\mathrm{MtM}}$ aggregation;

3. (Lagged market data) $\widehat{\mathrm{MtM}}^{incr}$ should be based on the same time, say 0, and initial condition X_0 (including, modulo calibration, market data), as $\widehat{\mathrm{MtM}}^{init}$. This condition ensures a consistent aggregation of $\widehat{\mathrm{MtM}}^{init}$ and $\widehat{\mathrm{MtM}}^{incr}$ into $\widehat{\mathrm{MtM}}^{augm}$.

These conditions have the following implications:

1. seems to ban second order generation XVAs, such as CVA in presence of initial margin, but these can in fact be included with the help of appropriate regression techniques;

2. implies to store the driver paths that were simulated for the purpose of obtaining $\widehat{\mathrm{MtM}}^{init}$; it also puts a bound on the accuracy of the estimation of MtM^{incr}, since the number of Monte Carlo paths is imposed by the initial run. Furthermore, the XVA desks may want to account for some wrong way risk dependency between the portfolio exposure and counterparty credit risk (see Sect. 4.2.1); approaches based on correlating the default intensity and the market exposure in (4.5) are readily doable in the present framework, provided the trajectories of the drivers and/or risk factors are shared between the clean and XVA desks;

3. induces a lag between the market data (of the preceding night) that are used in the computation of $\widehat{\mathrm{MtM}}^{incr}$ and the exact MtM^{incr} process; when the lag on market data becomes unacceptably high (because of time flow and/or volatility on the market), a full reevaluation of the portfolio exposure is required.

Figure 4.3.1 depicts the embedding of an MtM store-and-reuse approach into the trade incremental XVA engine of a bank.

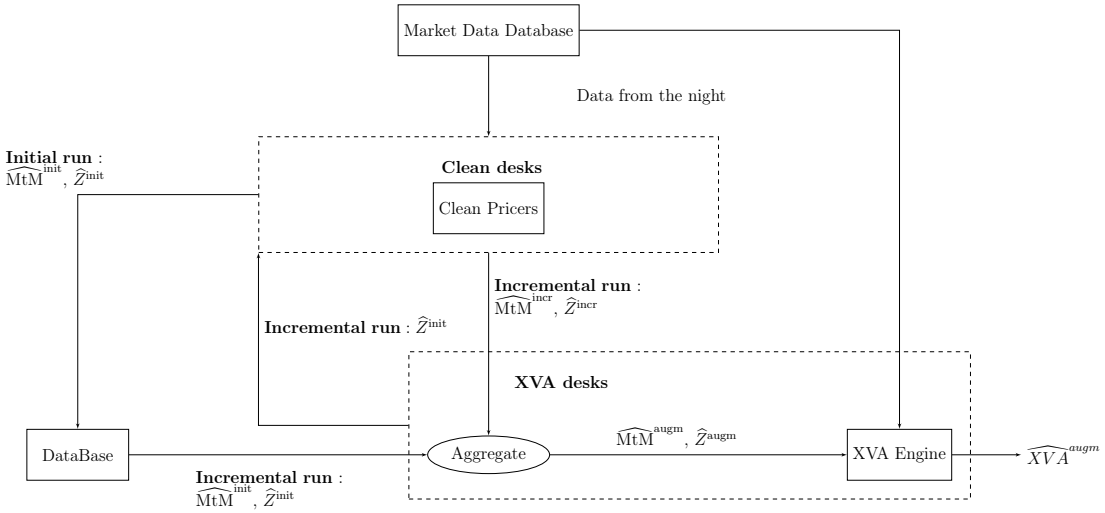


Figure 4.3.1: MtM store-and-reuse implementation of a trade incremental XVA engine with drivers Z.

4.3.2 Parallelization of the Genetic Algorithm

Most of the XVA compression computational time is spent in the evaluation of the incremental XVA metric involved in the fitness criterion visible in (4.8). The MtM store-and-reuse approach allows reducing the complexity of such trade incremental XVA computations to trade (as opposed to portfolio) size. However, in order to achieve XVA compression in real time, this is not enough; another key step is the parallelization of the genetic algorithm that is used for solving (4.8).

The genetic algorithm is a population based method, which implies to maintain a population of individuals (tentative new deals) through each iteration of the algorithm. The calculation of the objective function, for a given individual, does not depend on the fitness value of the other individuals. Therefore we can vectorize the computation of the fitness values within the population. Provided a suitable parallel architecture is available, a perfectly distributed genetic algorithm makes the execution time independent of the population size P (see Algorithm 4 and Figure 4.2.1).

This makes an important difference with other metaheuristic optimization algorithm, such as simulated annealing or stochastic hill climbing, which only evaluate one or very few solutions per iteration, but need much more iterations to converge toward a good minimum (see Adler (1993) and Ram, Sreenivas, and Subramaniam (1996)). As discussed in Pardalos, Pitsoulis, Mavridou, and Resende (1995), the above parallelization of the fitness function evaluation, for a given population, should not be confused with a parallel genetic algorithm in the sense of an independent evolution of several smaller populations.

In our context where individuals only represent incremental trades, a parallelization of population fitness evaluation is compatible with an MtM store-and-reuse approach for the trade incremental XVA computations. Combining the two techniques results in an XVA compression time independent of the sizes of the initial portfolio of the bank and of the population of the genetic algorithm used for the optimization, which represents an XVA compression computation time gain factor of the order of

$$\text{Number of trades in the initial portfolio} \times \text{population size}.$$

4.4 Case Study

In the remainder of the chapter, we present CVA compression results on real swap portfolios[3], using an additional swap for the CVA compression. We aim at addressing issues such as:

- Which type of swap is suitable for achieving the compression of the CVA, in the context of a given initial portfolio?
- How does the compression distort the portfolio exposure, with or without penalization?

To ease the implementation of the MtM store-and-reuse approach, we assume no CSA (cf. Sect. 4.3.1).

4.4.1 New Deal Parameterization

A swap is parameterized by its notional, its maturity, its direction, and its currency. The quantitative parameters are encoded through grids of values:

- Notional: from 10^5 to 10^7 by step of 10^5 dollars,
- Maturity: from 1 to 20 years by step of 1 year, 30 years and 50 years.

The qualitative parameters are encoded as enumerations of values:

- currency : Euro, US dollar, GBP or Yen.
- direction : A binary variable for payer or receiver.

[3]The underlying interest rate and FX models are proprietary and cannot be disclosed in the chapter. We use a deterministic credit spread model for the counterparty, calibrated to the CDS term structure of the latter.

Moreover we impose the additional swap to be at par so that it can be entered at no cost, which is equally desirable from the bank and the client perspectives.

The above parameterization defines a discrete search space $\mathcal{A}$ with $100 \times 22 \times 4 \times 2 = 1.76 \times 10^4$ elements.

4.4.2 Design of the Genetic Algorithm

We address the optimization problem (4.8) by a genetic algorithm as per Section 4.2.3. The new deal space $\mathcal{A}$ in (4.8) is viewed as a space of chromosomes x, the genes (deal parameters) of which evolve randomly along the iterations of the algorithm as detailed in Sect. 4.2.3.

In the theoretical literature on genetic algorithms, an individual is represented as a bit string . In practice, however, bit string representation of parameters does not give enough control on the mutation distribution. Namely, in bit string representation, mutations affect all bits uniformly, whereas we might want to mutate some parameters more frequently (the quantitative parameters, in particular, as the algorithm tends to quickly identify the relevant values of the qualitative parameters). Hence, we rather model our individuals x by a variable string, a choice also made in Kondratyev and Giorgidze (2017).

We choose rank proportionate selection to avoid fitness scaling issues. More precisely, if we have a population $\mathcal{P} = \{1, ..., P\}$ of P individuals and the associated fitnesses $(f_i)_{i \in \mathcal{P}}$, then the probability to select chromosome i is

$$p_i = \frac{2rank(f_i)}{P(P+1)},$$

where *rank* is a function that ranks chromosoms according to their fitness value (returning one for the highest value, in the context of a minimization problem).

Regarding the crossover operator, we use a uniform crossover mask, i.e. the choice of gene inherited from one parent or another is drawn with a uniform probability.

Regarding mutations, the probability to mutate a gene is proportional to the number of alleles (values) that it can take. The mutation operator then selects uniformly a new gene allele (value).

In our experiments, the notional of the new swap can take 100 different values, its currency 4 values, its position 2 values, its maturity 22 values. Hence, when a chromosome is selected for mutation, the probability to mutate each of its genes is equal to $\frac{100}{128}$ for the notional, $\frac{22}{128}$ for the maturity, $\frac{4}{128}$ for the currency, and $\frac{2}{128}$ for the position. Indeed , a more frequent mutation of notional and maturity parameters are desirable. Diversity for currency and position is ensured at the initialization of the algorithm (with a large population) and maintained across

the iterations thanks to the crossover operator. A prerequisite for a successful implementation is a reasonable specification of the search space $\mathcal{A}$.

Hyperparameters strongly impact the behavior of the algorithm. In the case of XVA compression, which is time-consuming, searching good values for the hyperparameters by a grid search method would be too demanding computationally. In our numerics, the mutation rate r_m is set to 20% and the crossover rate r_c to 50%. In the genetic algorithms literature the crossover rate is often close to one, but for problems with few genes (i.e. components of x, or parameters, only four in our case), it is recommended to select a smaller value.

With parallelization in mind (see Sect. 4.3.2), we prefer to decrease the number of iterations even if it implies to explore more solutions. We set the genetic algorithm population size to $P = 100$ individuals and we limit the number of iterations to 5. Hence, we value the fitness function on 600 tentative new swaps x.

4.4.3 Results in the Case of Payer Portfolio Without Penalization

First, we consider a portfolio only composed of payer swaps. The expected exposure (EE) and the expected positive and negative exposures (EPE and ENE), i.e. $\mathbb{E}\mathrm{MtM}_t$ and $\mathbb{E}\mathrm{MtM}_t^{\pm}$, are shown as a function of time t in Figure 4.4.1, which illustrates the asymmetric market risk profile of the portfolio.

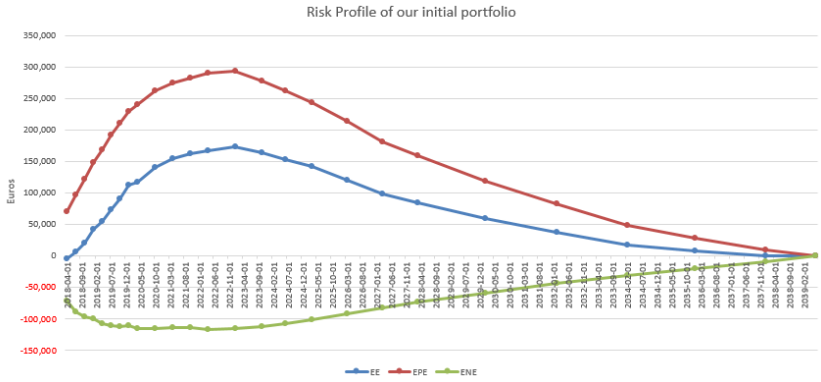


Figure 4.4.1: Market risk profile of the portfolio (payer portfolio without penalization)

Our first point is to verify that the algorithm without penalization, i.e. for $\alpha = 0$ in (4.8), will select a receiver swap with a maturity comparable to those of the swap of the initial portfolio.

Table 4.4.1 reports after each iteration the three best solutions (from top to bottom) ever found since the beginning of the algorithm (in terms of the fitness

criterion (4.8) with $\alpha = 0$, i.e. ΔCVA=). A negative incremental CVA means that the new swap decreases the counterparty risk of the bank. The initial portfolio CVA amounts to 34929€. We also report the |DV01|s of the augmented portfolios in order to be able to assess the impact of the penalization in our next experiment.

Iter.	Mat. (yrs)	Not. (K€)	Rate (%)	Curr.	Pos.	ΔCVA (€)	$\frac{-\Delta CVA}{CVA}$ (in %)	|DV01|(€)
0	10	4800000	1.6471	GBP	Receive	-8019	23.0	4484
	10	4700000	1.6471	GBP	Receive	-7948	22.8	4390
	10	4600000	1.6471	GBP	Receive	-7872	22.5	4297
1	17	5600000	1.4623	EUR	Receive	-17249	49.4	8648
	12	5400000	1.7036	GBP	Receive	-9163	26.2	5957
	16	3900000	0.6377	JPY	Receive	-8760	25.1	6137
2	14	6600000	1.3416	EUR	Receive	-21680	62.1	8626
	17	5100000	1.4623	EUR	Receive	-19729	56.5	7875
	17	5600000	1.4623	EUR	Receive	-17249	49.4	8648
3	14	6600000	1.3416	EUR	Receive	-21680	62.1	8626
	17	5100000	1.4623	EUR	Receive	-19729	56.5	7875
	17	5600000	1.4623	EUR	Receive	-17249	49.4	8648
4	17	3300000	1.4623	EUR	Receive	-27300	78.2	5096
	12	6100000	1.2203	EUR	Receive	-25382	72.7	6959
	11	5600000	1.147	EUR	Receive	-23009	65.9	5908
5	17	3300000	1.4623	EUR	Receive	-27300	78.2	5096
	12	6100000	1.2203	EUR	Receive	-25382	72.7	6959
	12	5100000	1.2203	EUR	Receive	-25264	72.3	5818

Table 4.4.1: Evolution of optimal solutions after each iteration (payer portfolio without penalization).

As seen in Figure 4.4.2, a stabilization of the algorithm is observed after 4 iterations, on a new swap leading to a CVA gain of about 27300€, i.e. about 78% of the initial portfolio CVA. The maturity and the notional are found the two most sensitive genes in the optimization. The maturity of the swap is chosen by the algorithm so as to reduce the exposure peak: The decrease of the exposure on the first 8 years of the portfolio is visible in terms of EPE profile on figure 4.4.3 and of CVA profile[4] on Figure 4.4.4.

4.4.4 Results in the Case of Payer Portfolio With Penalization

We keep the same initial portfolio but we now penalize our objective function by the |DV01| of the new swap, setting the regularization parameter α to one in (4.8). As will be seen below, this choice achieves a good balance between the two terms ΔCVA and αDV01 in (4.8).

[4]Term structure obtained by integrating the EPE profile against the CDS curve of the counterparty from time 0 to an increasing upper bound $t \leq T$ (cf. (4.3)).

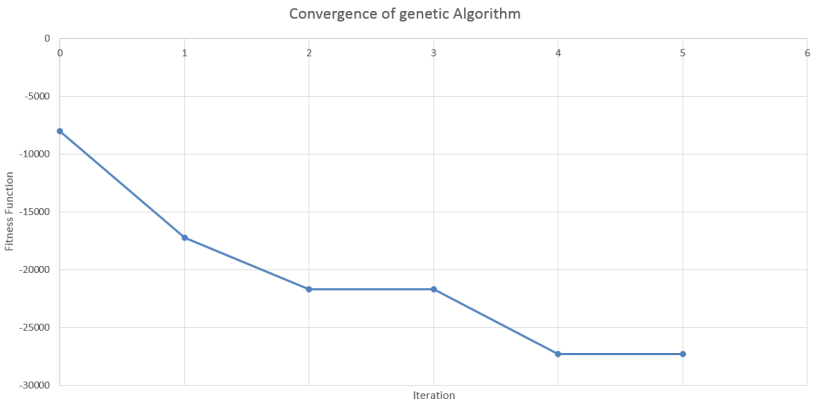


Figure 4.4.2: Fitness value as a function of iteration number (payer portfolio without penalization).

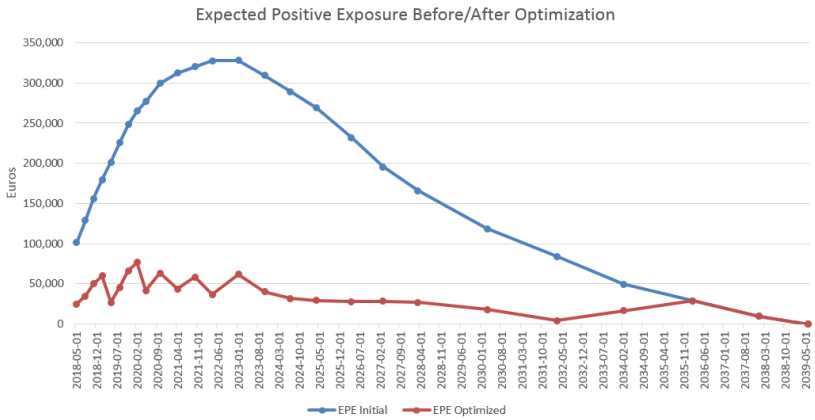


Figure 4.4.3: Market risk profile of the portfolio before and after optimization (payer portfolio without penalization).

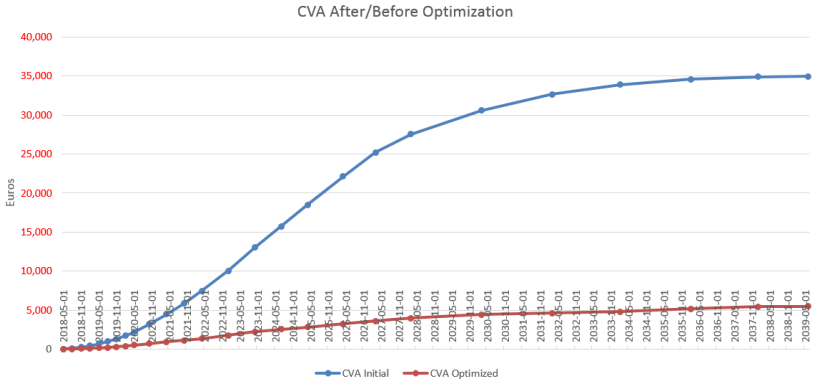


Figure 4.4.4: CVA profile before and after optimization (payer portfolio without penalization).

Iter.	Mat. (yrs)	Not. (K€)	Rate (%)	Curr.	Pos.	ΔCVA (€)	$\frac{-\Delta CVA}{CVA}$ (in %)	|DV01|(€)
0	10	4500000	1.6471	GBP	Receive	-7790	22.3	4218
	10	4600000	1.6471	GBP	Receive	-7871	22.5	4311
	10	4700000	1.6471	GBP	Receive	-7947	22.8	4405
1	17	5600000	1.4731	EUR	Receive	-16892	48.4	8706
	10	4500000	1.6471	GBP	Receive	-7790	22.3	4217
	10	4600000	1.6471	GBP	Receive	-7871	22.5	4311
2	14	6600000	1.3336	EUR	Receive	-21888	62.7	8654
	17	5600000	1.4731	EUR	Receive	-16892	48.4	8706
	17	6100000	1.4531	EUR	Receive	-15038	43.1	9466
3	14	6600000	1.3336	EUR	Receive	-21888	62.7	8654
	17	5600000	1.4731	EUR	Receive	-16892	48.4	8706
	9	4500000	0.9584	EUR	Receive	-10454	29.9	3945
4	10	6600000	1.3336	EUR	Receive	-21888	62.7	8654
	11	6600000	1.3999	EUR	Receive	-18825	53.9	9207
	17	5600000	1.4731	EUR	Receive	-16892	48.4	8706
5	11	2900000	1.3811	EUR	Receive	-25059	71.7	4039
	18	1500000	1.48	EUR	Receive	-18258	52.3	2442
	17	1500000	1.4531	EUR	Receive	-16553	47.4	2327

Table 4.4.2: Evolution of optimal solutions after each iteration (payer portfolio with penalization).

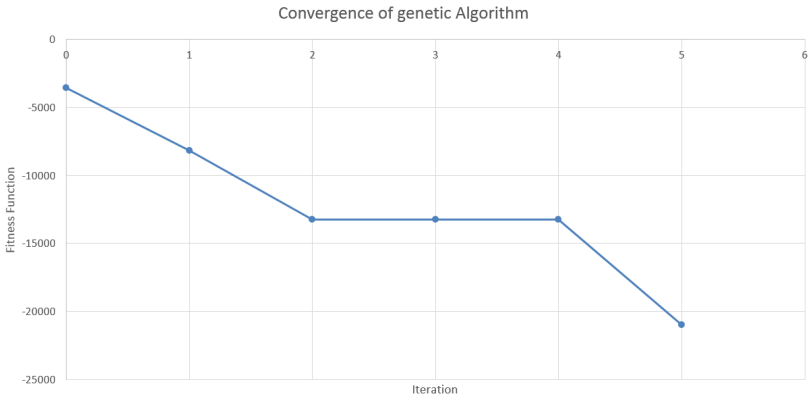


Figure 4.4.5: Fitness as a function of iteration number (payer portfolio with penalization).

In the present context of a payer portfolio, |DV01| control and CVA gain are two antagonistic targets. This may explain why the algorithm seems to struggle in finding a stable solution: indeed, the last iteration still decreases the fitness significantly (see Figure 4.4.5).

During the execution (see Table 4.4.2), the algorithm first optimizes the CVA and then (in iteration 5) reduces the |DV01|. This is due to the difference of order of magnitude between ΔCVA and |DV01| (recalling $\alpha = 1$): ΔCVA is more important, hence the algorithm only takes care of the penalization once ΔCVA has been compressed.

In the end, the gains in CVA are of the same order of magnitude as in the case without penalization (92% of the CVA gain without penalization), but for about 20% of |DV01| less than before. The second and third best solutions also achieve a great CVA gain, while diminishing the |DV01| by a factor three with respect to the nonpenalized case. By comparison with the unpenalized case (cf. Tables 4.4.1 and 4.4.2), the trades identified by the algorithm have a lower maturity or a smaller notional, hence a smaller |DV01|.

See Figures 4.4.6 and 4.4.7 for the corresponding market risk and CVA profiles before and after the optimization.

4.4.5 Results in the Case of a Hybrid Portfolio With Penalization

Next, we challenge our algorithm with a more balanced initial portfolio, as shown in Figure 4.4.8 (to be compared with Figure 4.4.1). The initial CVA is now 6410€. We set the regularization parameter α in (4.8)) to 0.3, as opposed to 1 in the

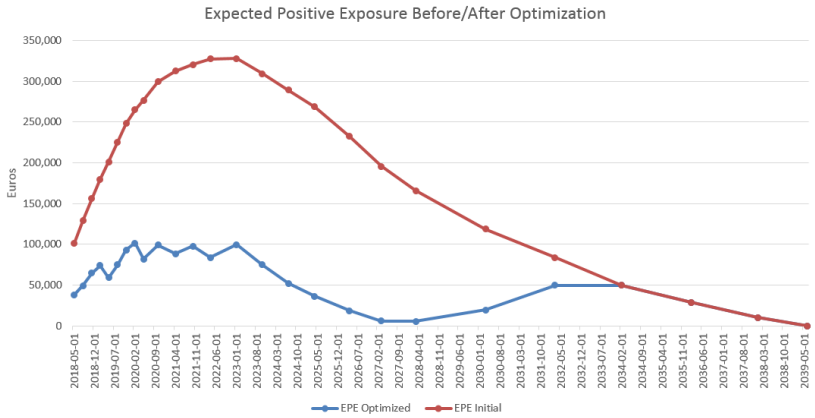


Figure 4.4.6: Market risk profile of portfolio before and after optimization (payer portfolio with penalization).

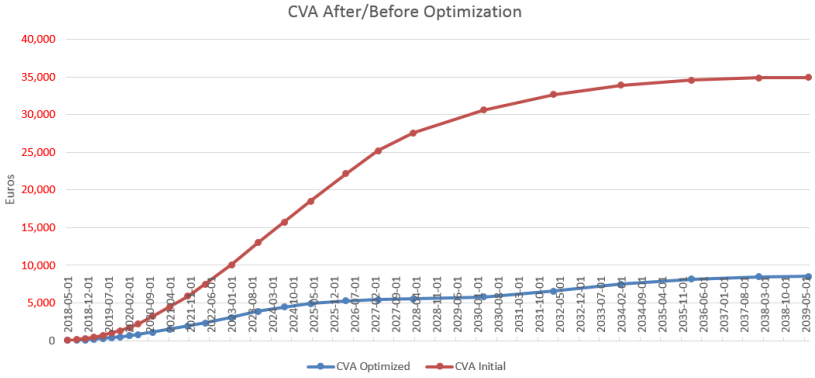


Figure 4.4.7: CVA profile before and after optimization (payer portfolio with penalization).

previous case, in view of the lower CVA of the initial portfolio.

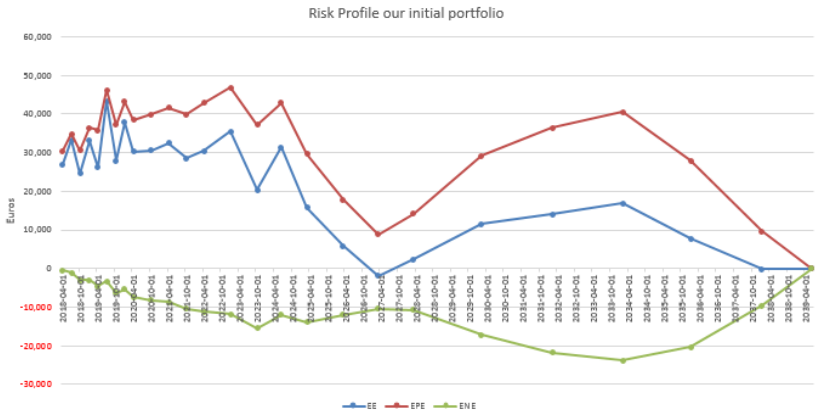


Figure 4.4.8: Market risk profile of the portfolio (hybrid portfolio with penalization).

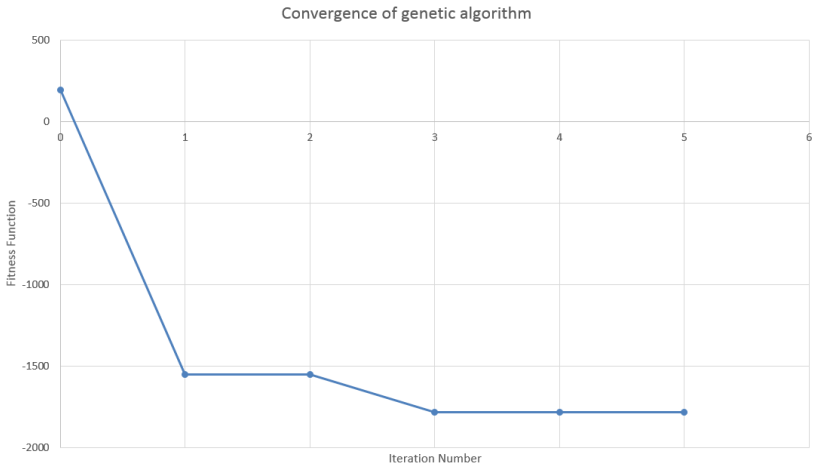


Figure 4.4.9: Fitness value as a function of iteration number (hybrid portfolio with penalization)

As visible in Figure 4.4.9, the stabilization of the algorithm occurs after three iterations, showing that, for the hybrid portfolio, |DV01| penalization and ΔCVA play less antagonistic roles. This is obtained by a relatively small notional and a maturity limited to 9 years, versus 11 years in the previous case of a payer portfolio with penalization. The corresponding market risk and CVA profiles, before and after the optimization, are displayed in Figures 4.4.10 and 4.4.11. Figure 4.4.10 explains the choices operated by the algorithm : As we restrict our incremental

Iter.	Mat. (yrs)	Not. (K€)	Rate (%)	Curr.	Pos.	ΔCVA (€)	$\frac{-\Delta CVA}{CVA}$ (in %)	|DV01|(€)
0	1	6000000	0.025	JPY	Receive	14	-0.2	609
	1	6100000	0.025	JPY	Receive	14	-0.2	619
	1	6300000	0.025	JPY	Receive	14	-0.2	640
1	8	1500000	0.8565	EUR	Receive	-1905	29.7	1177
	6	2300000	0.586	EUR	Receive	-1166	18.2	1370
	9	700000	1.608	GBP	Receive	-820	12.8	595
2	8	1500000	0.8565	EUR	Receive	-1905	29.7	1177
	6	2300000	0.586	EUR	Receive	-1166	18.2	1370
	9	700000	1.608	GBP	Receive	-82	12.8	595
3	9	1900000	0.9584	EUR	Receive	-2284	35.6	1665
	8	1500000	0.8565	EUR	Receive	-1905	29.7	1177
	7	2700000	0.7225	EUR	Receive	-1628	25.4	1865
4	9	1900000	0.9584	EUR	Receive	-2284	35.6	1665
	8	1500000	0.8565	EUR	Receive	-1905	29.7	1177
	7	2700000	0.7225	EUR	Receive	-1628	25.4	1865
5	9	1900000	0.9584	EUR	Receive	-2284	35.6	1665
	8	1500000	0.8565	EUR	Receive	-1905	29.7	1177
	9	2500000	0.9584	EUR	Receive	-1942	30.3	2192

Table 4.4.3: Evolution of optimal solutions after each iteration (hybrid portfolio with penalization).

strategy to one swap, the algorithm limits the EPE until the first positive peak before 2026. A better strategy, but one outside our search space $\mathcal{A}$, would be to add a second swap with entry date in 2028 and end date in 2037.

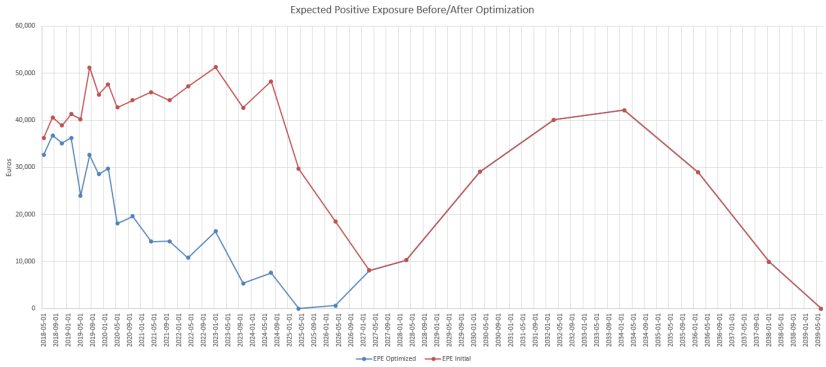


Figure 4.4.10: Market risk profile of portfolio before and after optimization (hybrid portfolio with penalization).

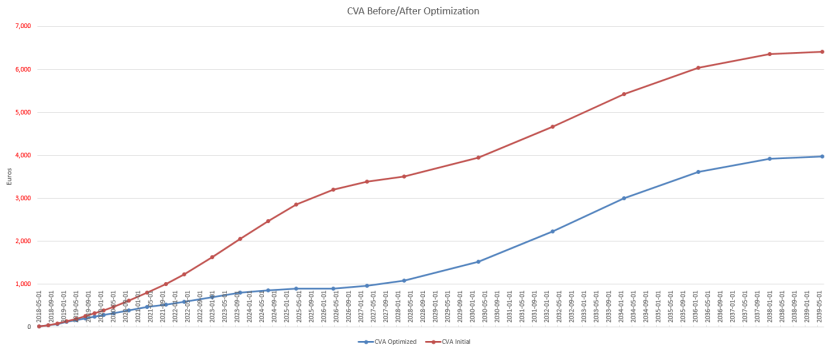


Figure 4.4.11: CVA profile before and after optimization (hybrid portfolio with penalization).

4.5 Conclusion

There exists a trade-off between CVA compression and DV01 penalization, which have antagonistic influences on the incremental exposure. Provided the search space for incremental trades is adequately chosen and parameterized, genetic optimization can result in significant CVA gains and, under DV01 penalization, this can be achieved without too much impact on the market risk of the bank position.

On the portfolios considered in our case studies, with ten to twenty trades, a basic XVA compression run on a standard PC without the acceleration techniques of Section 4.3 takes about 20 hours. The time gain resulting from an MtM store-and-reuse implementation of the trade incremental XVA computations as per Section 4.3.1 primarily depends on the size of the initial portfolio, but also on the maturity, and complexity more generally (vanilla vs. callable or path-dependent,...), of the constituting trades. Likewise, the time gain resulting from a parallel implementation of the genetic algorithm as per Section 4.3.2 primarily depends on the population size P, but it can be deteriorated by grid latency, hardware limitation, or data flow management, features. In our simulations, an MtM store-and-reuse implementation of the trade incremental XVA computations reduces the XVA compression time to about seven hours; A further parallel implementation of the genetic optimization algorithm lowers the execution time to about one hour.

The case study of this chapter is only a first step toward more complex optimizations. One could thus enlarge the search space with, e.g., crosscurrency swaps. In this case, the market risk penalization should be revisited to penalize other risk factors, beyond interest rate risk that is already accounted for by |DV01|. The penalization could also be refined with a focus on forward mark-to-market, i.e. market risk *in the future* (our current |DV01| penalization only controls spot

market risk).

CVA compression strategies involving several additional trades could be implemented. A first step toward such a multi-variate, multi-trade, compression would be an iterated application of single-trade XVA compressions, whereby, after each compression, the optimally augmented portfolio becomes the initial portfolio for the next compression. The benefit of such an iterative approach would be the ability to work with a search space $\mathcal{A}$ (or a sequence of them) of constant size, as opposed to a global search space $\mathcal{A}$ that would need to grow exponentially with the number of new trades in the case of a single multi-trade compression cycle.

Additional XVA metrics, and ultimately the all-inclusive XVA add-on (4.1), should be included in the compression (which, in particular, would allow one to identify possible XVA cuts across different netting sets).

Bibliography

Abbas-Turki, L. A., S. Crépey, and B. Diallo (2018). XVA Principles, Nested Monte Carlo Strategies, and GPU Optimizations. *International Journal of Theoretical and Applied Finance 21*, 1850030.

Ackerer, D., N. Tagasovska, and T. Vatter (2019). Deep smoothing of the implied volatility surface. *Available at SSRN 3402942*.

Adler, D. (1993). Genetic algorithms and simulated annealing: A marriage proposal. In *IEEE International Conference on Neural Networks*, pp. 1104–1109. IEEE.

Aggarwal, C. C. (2017). Outlier analysis second edition. Springer.

Albanese, C., M. Chataigner, and S. Crépey (2019). Wealth transfers, indifference pricing, and XVA compression schemes. In Y. Jiao (Ed.), *From Probability to Finance—Lecture note of BICMR summer school on financial mathematics*, Mathematical Lectures from Peking University Series. Springer. Forthcoming.

Albanese, C., S. Crépey, and M. Chataigner (2018). Wealth transfers, indifference pricing, and xva compression schemes. *From Probability to Finance - Lecture note of BICMR summer school on financial mathematics*.

Alfeld, P. (1984). A trivariate clough—tocher scheme for tetrahedral data. *Computer Aided Geometric Design 1*(2), 169–181.

Allouche, M., S. Girard, and E. Gobet (2021). Tail-gan: Simulation of extreme events with relu neural networks.

An, J. and S. Cho (2015). Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE 2*(1), 1–18.

Anandakrishnan, A., S. Kumar, A. Statnikov, T. Faruquie, and D. Xu (2018). Anomaly detection in finance: editors' introduction. In *KDD 2017 Workshop on Anomaly Detection in Finance*, pp. 1–7.

Armenti, Y. and S. Crépey (2017). Central clearing valuation adjustment. *SIAM Journal on Financial Mathematics 8*(1), 274–313.

Aubin-Frankowski, P.-C. and Z. Szabo (2020). Hard shape-constrained kernel machines. arXiv:2005.12636.

Azimi, M. and A. Agrawal (2019). Is positive sentiment in corporate annual reports informative? evidence from deep learning. *The Review of Asset Pricing Studies*.

Bachoc, F., A. Lagnoux, A. F. López-Lopera, et al. (2019). Maximum likelihood estimation for Gaussian processes under inequality constraints. *Electronic Journal of Statistics 13*(2), 2921–2969.

Bachouch, A., C. Huré, N. Langrené, and H. Pham (2021). Deep neural networks algorithms for stochastic control problems on finite horizon: numerical applications. *Methodology and Computing in Applied Probability*, 1–36.

Back, T. (1996). *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press.

Bartram, S. M., J. Branke, and M. Motahari (2020). *Artificial intelligence in asset management*. Number 14525. CFA Institute Research Foundation.

Basel Committee on Banking Supervision (2011, June). Basel iii: A global regulatory framework for more resilient banks and banking systems. Consultative document.

Basel Committee on Banking Supervision (2015). Review of the credit valuation adjustment risk framework. Consultative document.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pp. 437–478. Springer.

Bengio, Y., I. Goodfellow, and A. Courville (2017). *Deep Learning*, Volume 1. Citeseer.

Bengio, Y., P. Lamblin, D. Popovici, and H. Larochelle (2007). Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pp. 153–160.

Bergstra, J., R. Bardenet, Y. Bengio, and B. Kégl (2011). Algorithms for hyper-parameter optimization. *Advances in neural information processing systems 24*.

Blechschmidt, J. and O. G. Ernst. Three ways to solve partial differential equations with neural networks—a review. *GAMM-Mitteilungen*, e202100006.

Blickle, T. and L. Thiele (1995). A comparison of selection schemes used in genetic algorithms. TIK-Report.

Brigo, D. and F. Vrins (2018). Disentangling wrong-way risk: pricing credit valuation adjustment via change of measures. *European Journal of Operational Research 269*(3), 1154–1164.

Bühler, H., L. Gonon, J. Teichmann, and B. Wood (2019). Deep hedging. *Quantitative Finance 19*(8), 1271–1291.

Buhler, H., B. Horvath, T. Lyons, I. P. Arribas, B. Wood, et al. (2020). A data-driven market simulator for small data environments. Technical report.

Burgard, C. and M. Kjaer (2011). PDE Representations of Options with Bilateral Counterparty Risk and Funding Costs. *Journal of Credit Risk 7*(3), 1–19.

Cansado, A. and A. Soto (2008). Unsupervised anomaly detection in large databases using Bayesian networks. *Applied Artificial Intelligence 22*(4), 309–330.

Cappozzo, A., F. Greselin, and T. B. Murphy (2020). Anomaly and novelty detection for robust semi-supervised learning. *Statistics and Computing*, 1–27.

Carvalho, D. B., J. N. Bittencourt, and T. D. Maia (2011). The simple genetic algorithm performance: A comparative study on the operators combination. In *The First International Conference on Advanced Communications and Computation*.

Chaloner, K. and R. Brant (1988). A Bayesian approach to outlier detection and residual analysis. *Biometrika 75*(4), 651–659.

Chandola, V., A. Banerjee, and V. Kumar (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR) 41*(3), 1–58.

Chataigner, M., A. Cousin, S. Crépey, M. Dixon, and D. Gueye (2021). Beyond surrogate modeling: Learning the local volatility via shape constraints. *SIAM Journal on Financial Mathematics / Short Communications*. Forthcoming.

Chataigner, M., S. Crépey, and M. Dixon (2020). Deep local volatility. *Risks 8*(3), 82.

Chen, L., M. Pelger, and J. Zhu (2020). Deep learning in asset pricing. *Available at SSRN 3350138*. Working paper.

Chen, S.-H. (2012). *Genetic algorithms and genetic programming in computational finance*. Springer Science & Business Media.

Cousin, A., H. Maatouk, and D. Rullière (2016a). Kriging of financial term structures. *European Journal of Operational Research 255*, 631–648.

Cousin, A., H. Maatouk, and D. Rullière (2016b). Kriging of financial term-structures. *European J. Oper. Res. 255*(2), 631–648.

Crépey, S. (2002). Calibration of the local volatility in a trinomial tree using Tikhonov regularization. *Inverse Problems 19*(1), 91.

Crépey, S. (2003). Calibration of the local volatility in a generalized Black–Scholes model using Tikhonov regularization. *SIAM Journal on Mathematical Analysis 34*(5), 1183–1206.

Crépey, S. (2004). Delta-hedging vega risk? *Quantitative Finance 4*(5), 559–579.

Crépey, S. (2013). *Financial Modeling: A Backward Stochastic Differential Equations Perspective*. Springer Finance Textbooks.

Crépey, S., R. Hoskinson, and B. Saadeddine (2021). XVA analysis from the balance sheet. *Quantitative Finance 21*(1), 99–123.

Crépey, S. and S. Song (2016). Counterparty risk and funding: Immersion and beyond. *Finance and Stochastics 20*(4), 901–930.

Crépey, S. and S. Song (2017). Invariance properties in the dynamic gaussian copula model. *ESAIM: Proceedings and surveys 56*, 22–41.

Crépey, S. and M. Dixon (2020). Gaussian process regression for derivative portfolio modeling and application to CVA computations. *Journal of Computational Finance 24*(1), 47–81.

Del Moral, P. and F.-K. Formulae (2004). Genealogical and interacting particle systems with applications.

Drake, A. E. and R. E. Marks (2002). Genetic algorithms in economics and finance: Forecasting stock market prices and foreign exchange—a review. In *Genetic algorithms and genetic programming in computational finance*, pp. 29–54. Springer.

Dugas, C., Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia (2009). Incorporating functional knowledge in neural networks. *Journal of Machine Learning Research 10*(Jun), 1239–1262.

Dupire, B. (1994). Pricing with a smile. *Risk 7*, 18–20.

Dupont, L., O. Fliche, and S. Yang (2020). Governance of artificial intelligence in finance. *Banque De France*.

Engl, H., M. Hanke, and A. Neubauer (1996). *Regularization of Inverse Problems*. Kluwer.

Ferguson, R. and A. Green (2018). Deeply learning derivatives. *arXiv preprint arXiv:1809.02233*.

Fissler, T., J. F. Ziegel, and T. Gneiting (2016). Expected shortfall is jointly elicitable with value at risk-implications for backtesting. *Risk Magazine*.

Garcia, R. and R. Gençay (2000). Pricing and hedging derivative securities with neural networks and a homogeneity hint. *Journal of Econometrics 94*(1-2), 93–115.

Gatheral, J. (2004). A parsimonious arbitrage-free implied volatility parameterization with application to the valuation of volatility derivatives. *Presentation at Global Derivatives & Risk Management, Madrid*, 0.

Gatheral, J. (2011). *The volatility surface: a practitioner's guide*. Wiley.

Gatheral, J. and A. Jacquier (2014). Arbitrage-free SVI volatility surfaces. *Quantitative Finance 14*(1), 59–71.

Gatheral, J., T. Jaisson, and M. Rosenbaum (2018). Volatility is rough. *Quantitative finance 18*(6), 933–949.

Gençay, R. and M. Qi (2001). Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging. *IEEE Transactions on Neural Networks 12*(4), 726–734.

Glasserman, P. and L. Yang (2018). Bounding wrong-way risk in cva calculation. *Mathematical Finance 28*(1), 268–305.

Glorot, X. and Y. Bengio (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256.

Goix, N., A. Sabourin, and S. Clémençon (2015). On anomaly ranking and excess-mass curves. In *Artificial Intelligence and Statistics*, pp. 287–295.

Goix, N., A. Sabourin, and S. Clémençon (2017). Sparse representation of multivariate extremes with applications to anomaly detection. *Journal of Multivariate Analysis 161*, 12–31.

Goldberg, D. (1989). *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley.

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. MIT Press.

Green, A. (2015). *XVA: Credit, Funding and Capital Valuation Adjustments*. John Wiley & Sons.

Gregory, J. (2015). *The xVA Challenge: counterparty credit risk, funding, collateral and capital*. John Wiley & Sons.

Hamida, S. B. and R. Cont (2005). Recovering volatility from option prices by evolutionary optimization. *The Journal of Computational Finance*.

Hastie, T., R. Mazumder, J. D. Lee, and R. Zadeh (2015). Matrix completion and low-rank svd via fast alternating least squares. *The Journal of Machine Learning Research 16*(1), 3367–3402.

Hawkins, D. M. (1980). *Identification of outliers*, Volume 11. Springer.

Hinton, G. E., S. Osindero, and Y.-W. Teh (2006). A fast learning algorithm for deep belief nets. *Neural computation 18*(7), 1527–1554.

Holland, J. H. et al. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

Horvath, B., A. Muguruza, and M. Tomas (2019). Deep learning volatility. *Available at SSRN 3322085*.

Huge, B. N. and A. Savine (2020). Differential machine learning. *Available at SSRN 3591734*.

Hull, J. and A. White (2012). Cva and wrong-way risk. *Financial Analysts Journal 68*(5), 58–69.

Hutchinson, J. M., A. W. Lo, and T. Poggio (1994). A nonparametric approach to pricing and hedging derivative securities via learning networks. *The Journal of Finance 49*(3), 851–889.

Jin, Z., Z. Yang, and Q. Yuan (2019). A genetic algorithm for investment–consumption optimization with value-at-risk constraint and information-processing cost. *Risks 7*(1), 32.

Katona, Z., M. Painter, P. N. Patatoukas, and J. Zeng (2018). On the capital market consequences of alternative data: Evidence from outer space. In *9th Miami Behavioral Finance Conference*.

Kingma, D. P. and J. Ba (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Kiran, B. R., D. M. Thomas, and R. Parakkal (2018). An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos. *Journal of Imaging 4*(2), 36.

Knorr, E. M. and R. T. Ng (1998). Algorithms for mining distance-based outliers in large datasets. In *VLDB*, Volume 98, pp. 392–403. Citeseer.

Kondratyev, A. (2018). Curve dynamics with artificial neural networks. *Risk Magazine*, May. Preprint version available at https://ssrn.com/abstract=3041232.

Kondratyev, A. and G. Giorgidze (2017). Evolutionary algos for optimising MVA. *Risk Magazine*, December. Preprint version available at https://ssrn.com/abstract=2921822.

Kroha, P. and M. Friedrich (2014). Comparison of genetic algorithms for trading strategies. In *International Conference on Current Trends in Theory and Practice of Informatics*, pp. 383–394. Springer.

Lakhina, S., S. Joseph, and B. Verma (2010). Feature reduction using principal component analysis for effective anomaly–based intrusion detection on NSL-KDD.

Larranaga, P., C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic (1999). Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review 13*(2), 129–170.

LeCun, Y., B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel (1989). Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems 2*.

Leshno, M., V. Y. Lin, A. Pinkus, and S. Schocken (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks 6*(6), 861–867.

Li, M. and F. Mercurio (2015). Jumping with default: wrong-way risk modelling for cva. *Risk Magazine, November*.

Li, X. P., L. Huang, H. C. So, and B. Zhao (2019). A survey on matrix completion: Perspective of signal processing. *arXiv preprint arXiv:1901.10885*.

Liu, H., S. Shah, and W. Jiang (2004). On-line outlier detection and data cleaning. *Computers & chemical engineering 28*(9), 1635–1647.

Longstaff, F. A. and E. S. Schwartz (2001). Valuing american options by simulation: a simple least-squares approach. *The review of financial studies 14*(1), 113–147.

Lu, L., P. Jin, and G. E. Karniadakis (2019). Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*.

Ludkovski, M. and Y. Saporito (2020). Krighedge: Gaussian process surrogates for delta hedging. arXiv:2010.08407.

Maatouk, H. and X. Bay (2017). Gaussian process emulators for computer experiments with inequality constraints. *Math. Geosci. 49*(5), 557–582.

MacKay, D. J. and D. J. Mac Kay (2003). *Information theory, inference and learning algorithms*. Cambridge university press.

Markowitz, H. (1952). " portfolio selection". the journal of finance 7 (1): 77-91.

Masci, J., U. Meier, D. Cireşan, and J. Schmidhuber (2011). Stacked convolutional auto-encoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks*, pp. 52–59. Springer.

Mazumder, R., T. Hastie, and R. Tibshirani (2010). Spectral regularization algorithms for learning large incomplete matrices. *The Journal of Machine Learning Research 11*, 2287–2322.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

Murphy, K. (2012a). *Machine Learning: A Probabilistic Perspective*. MIT Press.

Murphy, K. P. (2012b). *Machine learning: a probabilistic perspective*. MIT press.

Nguyen, L. T., J. Kim, and B. Shim (2019). Low-rank matrix completion: A contemporary survey. *IEEE Access 7*, 94215–94237.

Ohn, I. and Y. Kim (2019, June). Smooth Function Approximation by Deep Neural Networks with General Activation Functions. *Entropy 21*(7), 627.

Omar, S., A. Ngadi, and H. H. Jebur (2013). Machine learning techniques for anomaly detection: an overview. *International Journal of Computer Applications 79*(2).

Pardalos, P. M., L. Pitsoulis, T. Mavridou, and M. G. Resende (1995). Parallel search for combinatorial optimization: Genetic algorithms, simulated annealing, tabu search and grasp. In *International Workshop on Parallel Algorithms for Irregularly Structured Problems*, pp. 317–331. Springer.

Patcha, A. and J.-M. Park (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer networks 51*(12), 3448–3470.

Pykhtin, M. (2012). General wrong-way risk and stress calibration of exposure. *Journal of Risk Management in Financial Institutions 5*(3), 234–251.

Ram, D. J., T. Sreenivas, and K. G. Subramaniam (1996). Parallel simulated annealing algorithms. *Journal of parallel and distributed computing 37*(2), 207–212.

Rios, L. M. and N. V. Sahinidis (2013). Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization 56*(3), 1247–1293.

Ro, K., C. Zou, Z. Wang, and G. Yin (2015). Outlier detection for high-dimensional data. *Biometrika 102*(3), 589–599.

Rocke, D. M. and D. L. Woodruff (1996). Identification of outliers in multivariate data. *Journal of the American Statistical Association 91*(435), 1047–1061.

Roper, M. (2010). Arbitrage free implied volatility surfaces. *https://talus.maths.usyd.edu.au/u/pubs/publist/preprints/2010/roper-9.pdf*.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review 65*(6), 386.

Ruf, J. and W. Wang (2020). Neural networks for option pricing and hedging: a literature review. *Journal of Computational Finance 24*(1).

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1985). Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Savine, A. (2018). *Modern Computational Finance: AAD and Parallel Simulations*. Wiley.

Schlegl, T., P. Seeböck, S. M. Waldstein, G. Langs, and U. Schmidt-Erfurth (2019). f-anogan: Fast unsupervised anomaly detection with generative adversarial networks. *Medical image analysis 54*, 30–44.

Schmidt-Hieber, J. (2020). Nonparametric regression using deep neural networks with relu activation function. *The Annals of Statistics 48*(4), 1875–1897.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell system technical journal 27*(3), 379–423.

Snoek, J., H. Larochelle, and R. P. Adams (2012). Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems 25*.

Sprenger, T. O., P. G. Sandner, A. Tumasjan, and I. M. Welpe (2014). News or noise? using twitter to identify and understand company-specific news flow. *Journal of Business Finance & Accounting 41*(7-8), 791–830.

Strub, F., R. Gaudel, and J. Mary (2016). Hybrid recommender system based on autoencoders. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pp. 11–16. ACM.

Strub, F. and J. Mary (2015). Collaborative filtering with stacked denoising autoencoders and sparse inputs. In *NIPS workshop on machine learning for eCommerce*.

Taarit, M. I. (2018). *Pricing of XVA adjustments: from expected exposures to wrong-way risks*. Ph. D. thesis, Université Paris-Est.

Tabassum, M. and K. Mathew (2014). A genetic algorithm analysis towards optimization solutions. *International Journal of Digital Information and Wireless Communications (IJDIWC) 4*(1), 124–142.

Tegnér, M. and S. Roberts (2019). A probabilistic approach to nonparametric local volatility. *arXiv preprint arXiv:1901.06021*.

Ting, J.-A., E. Theodorou, and S. Schaal (2007). A kalman filter for robust outlier detection. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1514–1519. IEEE.

Trolle, A. B. and E. S. Schwartz (2010, November). An empirical analysis of the swaption cube. Working Paper 16549, National Bureau of Economic Research.

Tschannen, M., O. Bachem, and M. Lucic (2018). Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*.

Turing, A. M. (2009). Computing machinery and intelligence. In *Parsing the turing test*, pp. 23–65. Springer.

Vaswani, S., F. Bach, and M. Schmidt (2019). Fast and faster convergence of sgd for over-parameterized models and an accelerated perceptron. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1195–1204. PMLR.

Verma, R. and P. Lakshminiarayanan (2006). A case study on the application of a genetic algorithm for optimization of engine parameters. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering 220*(4), 471–479.

Weinan, E., J. Han, and A. Jentzen (2017). Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics 5*(4), 349–380.

Xing, F., E. Cambria, and R. Welsch (2019). *Intelligent Asset Management*. Springer.

Young, S. R., D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton (2015). Optimizing deep learning hyper-parameters through an evolutionary algorithm. In *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*, pp. 4. ACM.

université PARIS-SACLAY

ÉCOLE DOCTORALE
de mathématiques
Hadamard (EDMH)

Titre: Quelques contributions de l'apprentissage statistique à la finance : Volatilité, nowcasting, compression de CVA.

Mots clés: finance quantitative, analyse XVA, apprentissage statistique

Résumé: La complexité des enjeux régulatoires a motivé l'usage intensif de l'apprentissage statistique en finance de marché. Cette thèse vise à démontrer les ponts possibles entre la finance quantitative et des outils récents en machine learning. On s'attache dans un premier temps à rendre compatible ces nouveaux outils que sont les réseaux de neurones avec des contraintes inhérentes à la valorisation de produits dérivés. Le respect de ces contraintes rend possible l'obtention de la volatilité locale dont nous évoluons la validité par le biais de backtests. Puis une variation des auto-encodeurs est introduite pour détecter des outliers et corriger des observations incomplètes si nécessaire. La nouveauté de cette approche réside dans sa capacité à traiter des données dont l'indexation varie au cours du temps. Enfin on optimise grâce à un algorithme génétique une mesure de risque de contrepartie, à savoir la CVA. On détaille au passage les défis opérationnels liés à la mise en oeuvre des routines de compression de métriques XVA.

Title: Some contributions of machine learning to finance : volatility, nowcasting, CVA compression.

Keywords: quantitative finance, XVA analysis, machine learning

Abstract: The complexity of regulatory issues has motivated the intensive use of machine learning in investment banking. This thesis dissertation aims at demonstrating the possible interactions between quantitative finance and recent machine learning tools. Firstly, we endeavor to reconcile some new tools, which are neural networks, with the constraints inherent in the valuation of derivative products. Respecting these constraints makes it possible to obtain local volatility which is validated through backtests. Then a variation of the autoencoders is introduced to detect outliers and correct incomplete observations if necessary. The novelty of this approach lies in its ability to deal with data whose indexation varies over time. Finally, we optimize a counterparty risk measure, namely the CVA, using a genetic algorithm. We detail the operational challenges related to the implementation of the XVA metric compression routines.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France