Utilisation des algorithmes GAN pour la génération de scénarios économiques

BOURBON Kelly, HABIB Taha, MOAYEDPOUR Quentin, RAMESH Brian

Sous la supervision de M. BRACH Loïc & M. TAIBI Hassane.

Référent pédagogique : M. FERMANIAN Jean-David

4 octobre 2024

Résumé

Introduits notamment en génération d'images et connus pour leur utilisation dans la création de deepfakes, les algorithmes de type Generative Adversarial Networks (GAN) présentent un intérêt en finance, notamment pour leur capacité à générer de nouvelles données à partir de données d'entraînement. En s'appuyant sur la littérature existante, l'objectif de ce sujet sera de tester le potentiel et les limites de l'utilisation des algorithmes GAN en apprenant sur des données de marchés et en générant des scénarios futurs possibles. L'idée sera ainsi de proposer une alternative à la combinaison relativement classique des modèles stochastiques et de la simulation Monte Carlo.

Mots clés : GAN, réseaux de neurones, séries temporelles

Abstract

Introduced notably in image recognition and known for their use in deepfake generation, Generative Adversarial Networks (GAN) algorithms are of interest in finance, notably for their ability to generate market scenarios. Drawing on existing literature, the idea of this topic is to test the potential and limits of using GAN algorithms by learning from market data and generating possible future scenarios. The aim will thus be to propose an alternative to the relatively classic combination of stochastic models and Monte Carlo simulations.

Key words : GAN, neural networks, time series

Remerciements

Nous souhaitons remercier M. Brach et M. Taibi pour leur suivi, leur disponibilité tout au long de l'année, leur accompagnement tout au long du projet, ainsi que leurs conseils précieux qui nous ont guidés. Nous les remercions également de nous avoir fait confiance pour mener ce projet. Ce projet constitue un point déterminant dans notre formation académique et dont nous avons pris goût à participer.

Table des matières

1 Introduction

Les données procurent une source d'informations remarquable dans tous les domaines, et leur intérêt en finance est indéniable. Les données financières sont généralement disponibles sous forme de séries temporelles et sont essentielles à toutes les formes d'analyses financières. En effet, les analyses numériques basées sur la simulation de données prennent une ampleur croissante. L'utilisation des données en analyse financière couvre une large variété de domaines comme l'allocation d'actifs, la gestion des risques et, plus généralement, la prévision économique pour simuler le comportement de certains aspects financiers dans le futur. De nombreux outils se basent sur des données historiques pour traiter ces divers domaines. Traditionnellement, des modèles stochastiques sont utilisés pour comprendre les distributions complexes des données et répondre à ces problématiques. L'avantage de ces modèles réside dans les calculs rapides en utilisant des formules dites fermées. Dans un monde où le machine learning connaît une importance croissante, accompagné d'une puissance de calcul qui se développe de manière exponentielle, les modèles se basant uniquement sur les données en elles-mêmes se révèlent être des outils intéressants. Ainsi, le but de cette étude est de se focaliser sur une méthode de génération de données populaire : les algorithmes de type Generative Adversarial Networks (GAN).

Introduits par Goodfellow et al. [12] en 2014, les algorithmes de type Generative Adversarial Networks (GAN) ont suscité un grand intérêt dans le domaine de la génération d'images et sont également connus pour leur utilisation dans la création de deepfakes. Cependant, leurs applications ne se limitent pas à la sphère de l'image : ils ont également un potentiel prometteur en finance. Dans ce contexte, notre projet vise à explorer le potentiel et les limites de l'utilisation de modèles GAN dans le domaine financier. Plus précisément, nous nous concentrerons sur leur capacité à générer des scénarios de marché en apprenant à partir de données réelles. Cette approche offre une alternative intéressante à la combinaison plus traditionnelle de modèles stochastiques et de simulations Monte Carlo. Les données utilisées pour cette étude proviennent principalement de marchés financiers, fournissant des informations sur les prix et les caractéristiques des actifs.

Pour guider notre démarche, nous nous appuierons sur la littérature existante et des études spécifiques portant sur l'utilisation de ces algorithmes en finance. Les données utilisées seront ensuite présentées. Enfin, nous présenterons les résultats issus de la mise en œuvre et des différentes analyses.

Ce mémoire vise à contribuer à la compréhension des GAN en tant qu'outils puissants pour la génération de scénarios financiers, tout en soulignant leurs avantages et leurs limites. Nous espérons que cette étude ouvrira de nouvelles perspectives pour l'analyse quantitative des marchés et la gestion des risques. Nos résultats sont également répertoriés sur notre GitHub. [1].

1. `https://github.com/QMoayedpour/statapp-hsbc`

2 Revue de littérature et présentation théorique

Dans le domaine de l'analyse financière, l'efficacité des modèles utilisant des séries temporelles dépend largement de la quantité et de la diversité des données disponibles. La capacité à générer différents scénarios s'avère très utile pour la prise de décisions et pour réduire l'incertitude. Dans le domaine de l'analyse financière, cette approche permet de visualiser les résultats dans divers contextes de marché, ce qui favorise une gestion optimale des risques. Les assureurs, par exemple, collectent les primes et estiment les coûts ultérieurement pour investir leurs actifs et couvrir leurs risques. Ces derniers ont besoin d'un cadre régulier pour la prévision de scénarios. Les modèles classiques de génération de données en finance reposent souvent sur des hypothèses fortes qui peuvent être remises en question.

2.1 Méthodes statistiques classiques

Les méthodes basées sur des modèles probabilistes pour la génération de données sont très utilisées dans le secteur financier mais aussi dans d'autres domaines impliquant des séries temporelles. Les modèles stochastiques et les simulations Monte Carlo ont longtemps été et sont toujours des piliers de l'analyse financière, offrant des outils pour évaluer les risques et les opportunités sur les marchés. Les modèles stochastiques (comme Black & Scholes [6]) reposent souvent sur des hypothèses de distribution de probabilité comme le mouvement brownien. Cependant, leur capacité à capturer les caractéristiques non linéaires des marchés, telles que les queues épaisses des distributions et les sauts de prix, reste limitée.

En parallèle, les simulations Monte Carlo [11] ont élargi les horizons de modélisation en permettant la génération de multiples scénarios futurs à partir de variables financières. Elles ont été particulièrement utiles pour évaluer la sensibilité des portefeuilles aux changements de paramètres et estimer les distributions de probabilité des résultats financiers.

Dans ces deux approches, des hypothèses simplificatrices sont nécessaires pour rendre les calculs réalisables. Les simulations Monte Carlo utilisent souvent des distributions paramétriques simples. Cependant, ces simplifications peuvent être à l'origine d'approximations inexactes en sous-estimant la complexité, notamment en ce qui concerne la volatilité.

Nous retrouvons également les modèles avec une approche économétrique comme les modèles ARMA, ARCH/GARCH qui reposent eux aussi sur des hypothèses fortes. Ces modèles peuvent également être utilisés pour génerer des relations multivariées.

Cependant, la littérature [4] montre que certaines hypothèses, comme la normalité des rendements, ne sont pas toujours vérifiées. En particulier, ces modèles ont généralement des difficultés à prédire les événements extrêmes.Les méthodes de machine learning entrent ainsi en jeu et gagnent en popularité grâce à leur capacité à apprendre directement sur les données, avec une justification généralement plus empirique que théorique.

2.2 Generative Adversarial Networks

Notre étude s'intéresse à une famille de modèles particulière : les Generative Adversarial Networks (GAN). Il convient de présenter le fonctionnement théorique des modèles GAN ainsi que ses variantes que nous utiliserons pour la génération de séries financières.

2.2.1 Modèle GAN

L'objectif des modèles Generative Adversarial Networks (GAN) est d'apprendre à générer des données dont la distribution est la plus proche possible des données d'entraînement du modèle.

Les GAN reposent sur un jeu adverse entre deux réseaux de neurones jusqu'à la convergence vers un équilibre. Cette méthode d'apprentissage est non supervisée : l'apprentissage se fait sans recours aux données labellisées. Les GANs nécessitent seulement une base de données dont on souhaite reproduire la distribution. Les GAN sont composés de deux sous-modèles principaux : un générateur (noté G) et un discriminateur (noté D). Le rôle du générateur est de créer des données artificielles qui ressemblent étroitement aux données réelles présentes dans le jeu de données d'apprentissage. Le générateur prend en entrée un bruit z, qui est généralement gaussien ou uniforme, et sort des données synthétiques. Le générateur vise à capturer la distribution des données observées $P_{data}(X)$ et cherche à tromper le discriminateur en générant des données de plus en plus réalistes. En revanche, le discriminateur est chargé de distinguer les données générées des données réelles. Ainsi, le discriminateur se concentre sur la probabilité conditionnelle $P(Y|X)$, avec Y la nature des données : réelle ou générée. La figure 1 représente la structure d'un modèle GAN pour la génération d'images.

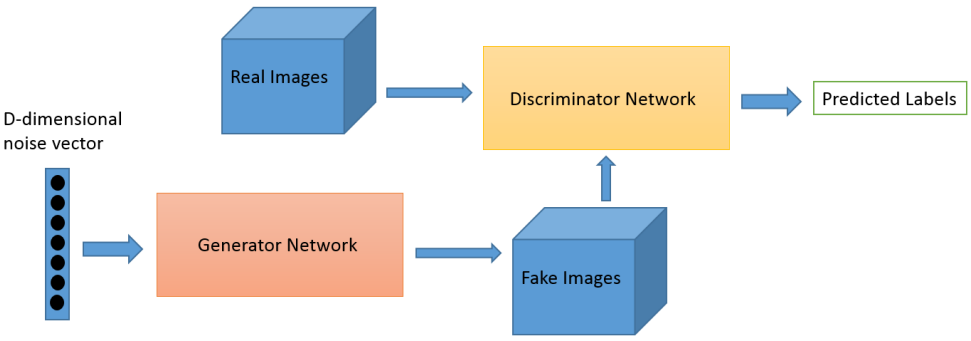


FIGURE 1 – Structure du GAN. Source : Oreilly

Formellement, on peut décrire la structure du GAN de base de la façon suivante : on note $z \sim P_z$, le bruit initial donné au générateur qui nous renvoie $G(z, \theta_g) \sim P_g$ où G est une fonction différentiable représentée par un réseau de neurones à plusieurs couches et de paramètres θ_g. Soit x l'input donné au discriminateur, on définit d'autre part une fonction $D(x, \theta_d)$ qui nous renvoie la probabilité que x appartient aux données réelles. On omet θ_d et θ_g dans la suite du rapport afin d'alléger les notations.

On observe donc que l'algorithme du GAN est un problème d'optimisation impliquant D et G que l'on peut mettre sous la forme : $\min_G \max_D V(G, D)$. Dans les GANs basiques cette fonction est donnée par :

$$V(D, G) = \mathbb{E}_{x \sim P_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim P_z(z)}[\log(1 - D(G(z)))]$$

Après avoir initialisé le générateur G et le discriminateur D avec des poids aléatoires, l'entraînement du GAN se fait de la manière suivante :

Algorithm 1 Algorithme du Réseau Antagoniste Génératif (GAN)

for $t = 1$ à T **do**
  for $k = 1$ à k_{train} **do**
    Echantillonner un minibatch de m échantillons de bruit $\{z^{(1)}, ..., z^{(m)}\}$ à partir de $P_z(z)$
    Echantillonner un minibatch de m données réelles $\{x^{(1)}, ..., x^{(m)}\}$ à partir de la distribution génératrice de données réelles $P_{data}(x)$
    Mettre à jour le discriminateur D à partir du gradient de sa fonction de perte :
      $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log D(x^{(i)}) + \log(1 - D(G(z^{(i)})))]$
  end for
  Echantillonner un minibatch de bruit $\{z^{(1)}, ..., z^{(m)}\}$ à partir de $P_z(z)$
  Mettre à jour le générateur G à partir du gradient de sa fonction de perte :
    $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(z^{(i)})))$
end for

Nous adoptons maintenant une approche idéaliste, similaire à celle décrite dans l'article de Goodfellow et al. [12], où le discriminateur et le générateur sont dotés d'une capacité infinie. Dans ce cadre, le problème minimax sous-jacent aux GAN est optimisé sur l'ensemble des fonctions. Ainsi, les réseaux de neurones du GAN ne sont pas restreints par leur aspect paramétrique. Les théorèmes suivants sont tirés du papier de Goodfellow et al. [12].

Théorème 1. *Pour G fixé, le discriminateur optimal est*

$$D_G^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_g(x)}$$

Ce résultat montre que le générateur vise à optimiser :

$$C(G) = \max_D \mathbb{E}_{x \sim P_{\text{data}}(x)} \left[\log \frac{P_{\text{data}}(x)}{P_{\text{data}}(x) + P_g(x)} \right] + \mathbb{E}_{z \sim P_z(z)} \left[\log \frac{P_{\text{data}}(x)}{P_{\text{data}}(x) + P_g(x)} \right]$$

Théorème 2. *Le minimum global du critère d'entraînement virtuel $C(G)$ est atteint si et seulement si $P_g = P_{data}$.*

Ce dernier théorème montre que la solution du problème minimax est donnée par $P_g = P_{\text{data}}$, c'est-à-dire que le générateur apprend la distribution des données d'entraînement tandis que le discriminateur est constant à $\frac{1}{2}$. Le théorème suivant montre que l'algorithme d'entraînement du GAN fait converger P_g vers P_{data}.

Théorème 3. *Si G et D ont une capacité suffisante, et à chaque étape de l'algorithme 1, le discriminateur est autorisé à atteindre son optimum étant donné G, et P_g est mis à jour de manière à améliorer le critère $C(G)$, alors P_g converge vers P_{data}.*

La littérature montre ensuite qu'il y a eu de nombreuses variations pour les différentes applications en finance [10]. Ces variations incluent la modification de l'architecture des réseaux de neurones, de la fonction de perte, de la structure des données, etc. Dans le cadre de notre projet, nous explorons les contributions potentielles de la génération de données financières à l'aide de modèles GAN et évaluons les structures plus (ou moins) pertinentes. Les parties suivantes présenteront l'aspect théorique de ces variations ainsi que leurs apports ou différences par rapport au cadre décrit ci-dessus. Les résultats associés à chaque méthode seront examinés dans la dernière partie.

2.2.2 Long Short-Term Memory (LSTM)

Les réseaux de neurones à mémoire à court et long terme, long short-term memory abrégé LSTM, font partie de la classe des réseaux neuronaux récurrents (RNN). Leur conception vise à capturer les dépendances séquentielles dans les données en utilisant une mémoire interne, ce qui les rend particulièrement adaptés à l'analyse des données séquentielles, telles que les séries temporelles. Contrairement aux réseaux de neurones feedforward où l'information circule dans un seul sens, les RNN fonctionnent selon une boucle : la sortie dépend à la fois de l'entrée courante et des entrées précédentes.

Cependant, les RNN rencontrent généralement des problèmes de vanishing gradient lorsqu'ils tentent de modéliser des dépendances à long terme. Cela signifie que l'ajustement des poids, notamment lorsqu'elle implique des éléments distants dans la séquence, peut converger rapidement vers zéro (ou dans certains cas, exploser), ce qui rend le réseau incapable de prendre en compte les entrées passées.

Introduits par Schmidhuber et Hochreiter en 1997 [15], les réseaux LSTM surmontent cette limitation en intégrant des cellules de mémoire, des portes et des connexions soigneusement conçues. Cette architecture leur permet de conserver et de propager sélectivement des informations sur des intervalles de temps prolongés. Les modèles LSTM sont ainsi capables de saisir des relations temporelles complexes dans des données séquentielles, ce qui les rend particulièrement efficaces pour la prédiction de séries temporelles telles que les cours boursiers.

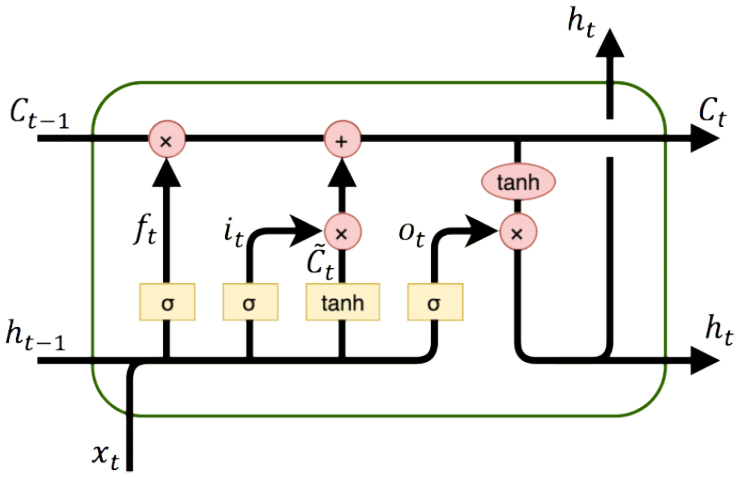


FIGURE 2 – Cellule de type LSTM

Comme nous pouvons le constater, une cellule LSTM se compose de trois portes caractérisant trois étapes distinctes : la porte d'oubli (*forget gate*) correspond à ce qui doit être conservé ou retiré de la cellule précédente, la porte d'entrée (*input gate*) correspond à l'information actuelle à intégrer, et enfin la porte de sortie (*output gate*) détermine quelle information doit être conservée en sortie, en fonction des informations actuelles et précédemment mémorisées. Nous notons C_t la mémoire de la cellule à l'instant t, h_t la valeur d'activation à cet instant, σ une fonction sigmoïde, W_j une matrice de poids (où l'indice j est spécifique à chaque calcul/étape), et b un vecteur de biais.

Le processus se passe en trois étapes principales à l'instant t :

1. Oubli : Calcul de $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$.
   La première étape de notre LSTM consiste à décider des informations que nous allons éliminer de l'état de la cellule. Cette décision est prise par une couche sigmoïde appelée "couche de porte d'oubli". Elle examine h_{t-1} et x_t, et produit un nombre entre 0 et 1 pour évaluer l'importance de chaque élément dans l'état de la cellule C_{t-1}. Un 1 représente "conserver complètement" tandis qu'un 0 représente "le retirer".
2. Ajout information : La deuxième étape consiste à décider des nouvelles informations que nous allons stocker dans l'état de la cellule. Cela se divise en deux parties. Tout d'abord, une couche sigmoïde appelée "couche de porte d'entrée", i_t, décide des valeurs que nous allons mettre à jour. Ensuite, une couche tanh crée un vecteur de nouvelles valeurs candidates, $\tilde{C}_t$, qui pourraient être ajoutées à l'état. Mathématiquement, la porte d'entrée est :
   $$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
   $$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$
   Dans l'étape suivante, nous allons combiner ces deux éléments pour créer une mise à jour de l'état.
3. Output : Enfin, nous construisons la mémoire de la cellule en calculant le nouvel état de la cellule C_t.
   Enfin, nous devons décider de ce que nous allons produire en sortie. Cette sortie sera basée sur notre état de cellule, mais sera une version filtrée. Tout d'abord, nous passons par une couche sigmoïde qui décide quelles parties de l'état de la cellule nous allons produire en sortie. Ensuite, nous appliquons la fonction tangente hyperbolique (tanh) à l'état de la cellule (pour faire en sorte que les valeurs soient comprises entre -1 et 1) et nous le multiplions par la sortie de la porte sigmoïde, afin de ne conserver que l'information importante.
   Formellement, la formule pour la porte de sortie est :
   $$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$
$$h_t = o_t \cdot \tanh(C_t)$$

Les cellules LSTM peuvent être utilisées dans l'architecture du générateur et discriminateur dans un modèle GAN. Ces dernières semblent appropriées pour un problème concernant des données temporelles.

2.2.3 Deep-Convolutional GAN (DCGAN)

Les DCGAN, ou Réseaux Antagonistes Génératifs Convolutions Profonds, représentent l'une des architectures de réseau les plus populaires et réussies pour les GANs. Introduits par Alec Radford, Luke Metz et Soumith Chintala [1], ils se distinguent par leur utilisation de réseaux de neurones convolutionnels à la fois pour le générateur et le discriminateur, ce qui leur permet de capturer des caractéristiques spatiales plus complexes et d'obtenir des résultats de meilleure qualité.

Le générateur G est défini par une série de couches de neurones convolutifs transposés, tandis que le discriminateur D est un réseau de neurones convolutif classique.

Un des principes clés des DCGAN est le remplacement des couches de pooling par des convolutions à pas (stride) dans le discriminateur et des convolutions transposées (fractional-strided convolutions) dans le générateur.

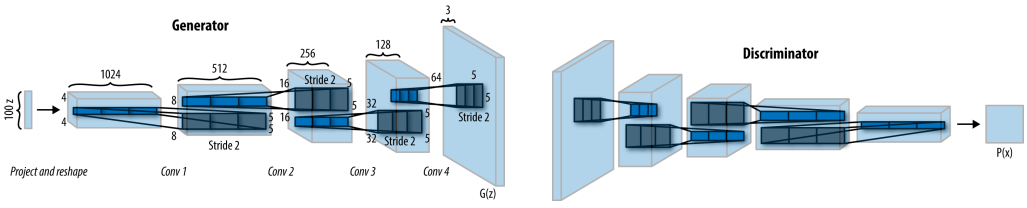
Formellement, le générateur génère, à partir d'un vecteur aléatoire, une fausse donnée en utilisant des convolutions transposées, ce qui pourrait être assimilé à un réseau convolutif inversé pour augmenter la dimension de notre vecteur d'entrée (upsampling). Les fonctions d'activation principales sont les Rectified Linear Unit (ReLU) dans toutes les couches, sauf la dernière où une fonction tanh est utilisée pour garantir des entrées normalisées pour le discriminateur.

Le discriminateur, quant à lui, effectue du downsampling pour réduire progressivement les dimensions des représentations de nos données. Un graphe qui détaille cela est en annexe (6)

Les DCGAN utilisent également la normalisation par lots (batch normalization) dans les deux réseaux. Cette technique stabilise l'apprentissage en normalisant l'entrée de chaque unité pour avoir une moyenne nulle et une variance unitaire, ce qui aide à résoudre les problèmes d'initialisation médiocre et facilite le flux de gradient dans les modèles plus profonds. Toutes ces innovations contribuent à stabiliser l'entraînement des DCGANs par rapport aux GANs traditionnels, ce qui permet d'obtenir des résultats de meilleure qualité et d'accélérer le processus d'apprentissage.

Les DCGANs sont généralement utilisés pour générer des données de haute dimension telles que des images, mais ils peuvent également être adaptés pour fonctionner avec des données unidimensionnelles comme dans notre cas pour des séries temporelles.

Un DCGAN 1D se compose d'un réseau de discrimination convolutionnel 1D et d'un réseau de génération convolutionnel transposé 1D. Le discriminateur prend une entrée 1D et produit une valeur scalaire, tandis que le générateur prend un vecteur de bruit aléatoire en entrée et génère une sortie 1D visant à tromper le discriminateur.



(a) Cours de l'actif (b) Log Returns

FIGURE 3 – Générateur du DC-GAN. Source : Oreilly

2.2.4 Wasserstein GAN (WGAN)

Introduit par Arjovsky et al. [2], le WGAN est une alternative à la structure traditionnelle des GANs.

Contrairement aux GANs classiques, où la divergence de Kullback-Leibler (KL) est souvent utilisée pour mesurer la différence entre les distributions de probabilité réelle et générée, le WGAN utilise la distance de Wasserstein, également appelée distance de Earth Mover (EM), pour quantifier cette différence. Cette approche rend l'entraînement plus stable et permet d'éviter le problème du mode collapse, qui peut parfois restreindre la variété des résultats proposés.

Le mode collapse se réfère à un problème rencontré lors de l'entraînement des GANs, où le générateur produit constamment les mêmes sorties ou des variations minimes de celles-ci. Cela indique que le modèle n'a appris qu'une partie restreinte de la distribution souhaitée, et reste bloqué dans un sous-espace limité de l'espace des données. Par exemple, dans le cas des données MNIST, qui comprennent des images de dix chiffres différents de "0" à "9", le mode collapse se manifeste lorsque le générateur ne génère qu'un ensemble d'images représentant le même chiffre, sans être capable de produire des images représentant l'ensemble des dix chiffres de manière variée.

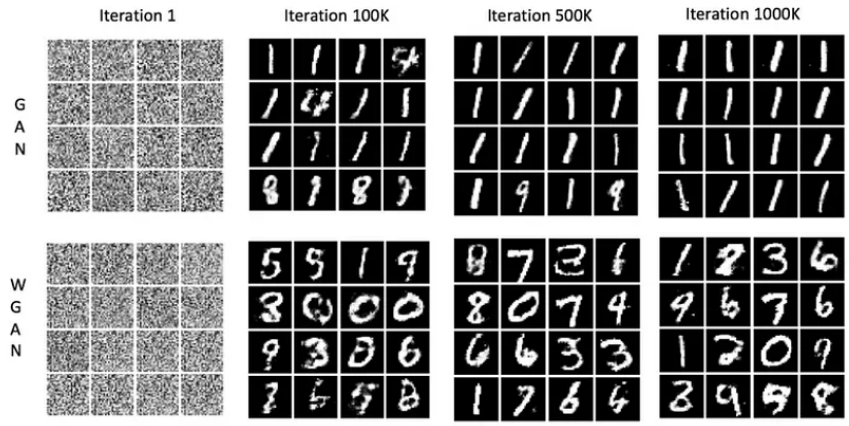


FIGURE 4 – Mode Collapse : exemple MNIST [19]

L'utilisation de la distance de Wasserstein permet au discriminateur d'identifier et de rejeter les éléments générés auxquels le générateur pourrait se stabiliser, permettant ainsi au générateur de diversifier ses générations.

Afin de mieux comprendre les améliorations de cette approche, il faut noter que le GAN original [12] minimise la divergence de Jensen-Shannon (JS) entre la distribution des données réelles et la distribution des données générées. Notons que la distance JS qui est optimisée par l'algorithme GAN basique n'est pas tout le temps continue, tandis que la distance de Wasserstein se caractérise par sa continuité et sa différentiabilité presque partout, ce qui permet d'entraîner le modèle jusqu'à l'optimalité. De plus, la distance de Wasserstein est une mesure significative, c'est-à-dire qu'elle converge vers 0 lorsque les distributions se rapprochent et diverge lorsqu'elles s'éloignent. Nous définissons formellement ces deux distances

dans l'annexe(6). Ces caractéristiques de la distance de Wasserstein offrent des meilleures propriétés de convergence que les autres distances classiques utilisées. Elle représente en quelque sorte le coût minimum de passage d'une distribution à une autre. Elle garantit une convergence relativement plus lisse d'autant plus lorsque les distributions sont assez proches.

En pratique dans le WGAN, nous utilisons une légère transformation de cette distance, car nous ne pouvons pas calculer l'infimum sur tous les $\gamma \in \Pi(P_r, P_s)$ où $P_r, P_s \in Prob(B)$ les deux distributions et B une métrique compacte. Cependant grâce à la dualité de Kantorovich-Rubinstein, nous avons :

$$W(P_r, P_s) = \sup_{f:X\to\mathbb{R} \text{ 1-Lipschitz}} \mathbb{E}x \sim P_r[f(x)] - \mathbb{E}x \sim P_s[f(x)] \tag{1}$$

où $W(P_r, P_s)$ est le supremum sur toutes les fonctions 1-Lipschitz $f : X \to \mathbb{R}$

Ainsi le nouveau problème de min-max du GAN devient : [3]

$$\min_G \max_{D\in T} \mathbb{E}_{x\sim p_{\text{data}}}[D(x)] - \mathbb{E}_{z\sim p_z}[D(G(z))]$$

où T est l'ensemble des fonctions 1-Lipschitzienne.

Dans le papier du WGAN, le discriminateur est appelé le critique. En effet, le discriminateur n'est plus un classifieur qui contrôle l'authenticité des données d'entrée par une estimation probabiliste. Au lieu de cela, il est entrainé à prédire une distance minimale de Wasserstein entre la distribution des données d'entraînement et celle des exemples générés en optimisant la fonction D :

$$\max_{D\in T} \mathbb{E}_{x\sim p_{\text{data}}}[D(x)] - \mathbb{E}_{z\sim p_z}[D(G(z))]$$

Bien que le WGAN améliore la stabilité de l'entraînement, il peut parfois ne pas converger. Les problèmes avec le WGAN proviennent principalement de la méthode de clipping des poids utilisée pour imposer la continuité de Lipschitz à la fonction D du critique. Les auteurs restreignent les poids w du réseau de neurones du discriminateur à un espace compact en les contraignant à se situer dans une plage restreinte ([-1e-2, 1e-2] dans l'article).

Dans notre étude, nous utilisons une version légèrement différente du WGAN, appelée WGAN-Gradient Penalty (WGAN-GP), qui remplace le clipping des poids par une contrainte sur la norme du gradient du critique pour imposer la continuité de Lipschitz. Cela permet un entraînement plus stable du réseau que le WGAN et nécessite très peu d'ajustement des hyperparamètres. Ce GAN est basé sur le papier original de Arjovsky et al. [2] et amélioré par Gulrajani et al. [13]. L'idée repose sur le théorème suivant : une fonction différentiable f est 1-Lipschitzienne si et seulement si ses gradients ont une norme inférieure ou égale à 1 partout. La méthode de la pénalité de gradient impose ainsi une contrainte telle que les gradients de la sortie du critique par rapport aux entrées aient une norme unitaire.

Le nouvel objectif du GAN devient donc :

$$\mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] - \mathbb{E}_{z\sim p_z(z)}[\log(D(G(z)))] + \lambda \mathbb{E}_{\hat{x}\sim p_{\hat{x}}}[(||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2]$$

Dans cette équation :

— $\hat{x}$ est un échantillon interpolé entre des échantillons réels et générés.

— $P_{\hat{x}}$ est la distribution de ces échantillons interpolés.

— λ est un coefficient de régularisation.

— $\nabla_{\hat{x}} D(\hat{x})$ est le gradient de la sortie du discriminateur par rapport à l'entrée interpolée $\hat{x}$.

$\mathbb{P}_{\hat{x}}$ est la distribution obtenue en échantillonnant uniformément le long d'une ligne droite entre les distributions réelles et générées $\mathbb{P}_{data}$ et $\mathbb{P}_g$.

Cette fonction de perte intègre la divergence de Wasserstein entre les distributions réelles et générées, ainsi qu'une pénalité de gradient qui garantit que le discriminateur satisfait la condition de Lipschitz.

3 Matériels et méthodes

Le but de ce papier est de génerer des scénarios économiques. En effet, on veut pour un actif $(S_t)_{t\in\mathbb{N}}$ observé, générer une série $(\tilde{S}_h, \tilde{S}_{h+1}..., \tilde{S}_{h+k})$ de variables dont la distribution est proche de $(S_h, S_{h+1}, \ldots, S_{h+k})$.

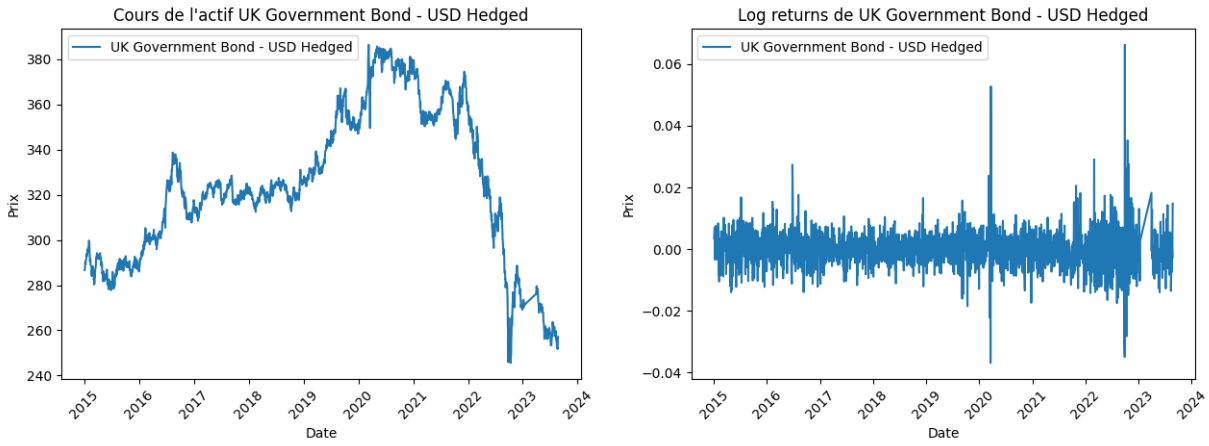
3.1 Description de nos données

Pour notre étude nous utilisons, une base de données composée de prix d'actifs financiers. Notre dataframe est composé de 28777 observations, 12 actifs financiers différents, ainsi que 2000 à 2500 observations par actif. Notre sujet étant focalisé sur la génération de séries, nous présenterons par la suite les résultats sur une série. Il est à noter que le choix de la série n'a pas d'importance ici et que les méthodes utilisées et les résultats observés (notamment les faits stylisés qui seront rappelés par la suite dans la section 3.2) sont en général valables pour toutes. Nous entraînons les modèles à générer les log returns des séries. Soit r_t le log return de l'actif à l'instant t qui est défini par $r_t = \ln\left(\frac{S_t}{S_{t-1}}\right)$. L'avantage d'utiliser le log return de la série plutôt que la valeur des actifs directement est que celui-ci est une approximation du taux de croissance de la série et qu'on peut ensuite vérifier si les données générées vérifient des faits stylisés sur le taux de rendement des actifs ([9], [8]). De plus, on remarque graphiquement [Figure 5] que le prix de l'actif ne semble clairement pas stationnaire [2]on entend ici la stationnarité au sens faible dont la définition est rappelée en annexe 6), tandis que le log return semble avoir une distribution plus stable dans le temps. Ce détail a un impact pratique lors de l'élaboration du modèle qui sera développé en section 3.3.

3.2 Propriétés des séries financières

Les séries financières présentent des propriétés particulières qui sont propres à la nature irrégulière des marchés financiers [9]. Cette partie vise à les présenter de manière générale. Nous les vérifierons sur notre base de données et sur nos séries générées dans la

2. (



(a) Cours de l'actif

(b) Log Returns

FIGURE 5 – Actif UK Government Bond - USD Hedged depuis 2015

section 4.3. Cela constituerait une base pour voir l'efficacité des modèles et déduire que les scénarios qui respectent ces conditions seraient plus réalistes.

1. Le premier fait concerne la stationnarité des séries. Empiriquement, on observe que la série des prix est non-stationnaire mais que la série des rendements l'est. Les prix des actifs financiers, peuvent souvent présenter des tendances à long-terme, des cycles et des volatilités importantes (comme nous pouvons le voir figure 5). Ces caractéristiques font que les séries de prix ne sont généralement pas stationnaires.
2. Les rendements ne sont pas autocorrélés mais les rendements en valeur absolus (et au carré) le sont.
3. Les séries de rendements sont caractérisées par des distributions à queue épaisse et avec un pic autour de la moyenne.
4. L'effet levier, expliqué par Bouchaud et Potters [7] représente la corrélation négative entre les rendements passés et les volatilités futures des actifs. La volatilité a tendance à augmenter en réponse à des rendements nets moins élevés que prévu et à chuter dans le cas inverse.
5. Le phénomène de cluster volatily, ou regroupement des volatilités est introduit par Mandelbrot(1963). On observe sur les marchés financiers que les chocs de volatilité se propagent. Ce regroupement des volatilités s'explique par les corrélations des séries temporelles. En effet, un grand mouvement de prix d'actif a alors plus de chance d'être suivi d'un mouvement de même ampleur, dans le même sens ou non.

Il existe encore d'autres faits qui sont suivis par les séries financières et qui donnent un caractère unique à celles-ci.

3.3 Entrainement des modèles

Afin d'entraîner nos modèles, on considère une série de log return d'un actif donné r et on en extrait des sous-échantillons $(r_i, r_{i+1}..., r_{i+k})$ où k représente la taille de la fenêtre

des échantillons. On choisit arbitrairement l'actif *UK Government Bond - USD Hedged* et on fixe k à 80. On dispose ainsi de 2395 échantillons de données. Pour tout $t \in \{80, 101...2315\}$, l'observation r_t apparaît dans 80 échantillons (mais à un index différent à chaque fois). Le faible nombre de données nous contraint à créer le plus de sous-échantillons possibles et nous avons observé de meilleurs résultats lorsque le nombre d'échantillons était plus élévé (mais en ayant des observations qui apparraissent dans plusieurs échantillons).

Afin d'entraîner les modèles à se situer dans la série, on considère que les 20 premières données d'entrée du générateur sont 20 données réelles, le reste étant un bruit gaussien. Cela permet de générer un scénario à partir de 20 données de départ. Cette modification a essentiellement pour apport de pouvoir guider en partie le modèle pour la génération des 60 données suivantes. Empiriquement, nous avons observé que cela permettait de mieux reproduire la saisonnalité de la série réelle.

Ensuite, on définit le générateur et le discriminateur. Pour chaque modèle testé, l'architecture du générateur et du discriminateur sont semblables : à l'exception de la dimension d'entrée et de sortie des deux modèles, les réseaux de neurones utilisés sont les mêmes pour chaque modèle.

Les modèles GAN sont des modèles non-supervisés et il peut être compliqué de définir un critère d'arrêt pour l'entraînement du modèle. En effet, ce n'est pas parce que le modèle aboutit à un discriminateur qui ne distingue plus les données réelles et générées, que les données générées sont pour autant vraisemblables. Dans le cas de la génération d'image, il est plûtot simple d'évaluer si une image générée est réaliste. Pour des séries temporelles, il faut définir une métrique pour évaluer nos données. On définit donc à chaque epoch [3] le score du modèle par :

$$score = |\tilde{M} - M| + \lambda * |\tilde{V} - V|$$

où :

— $\tilde{M}$ (resp. M) est la moyenne calculée sur les scénarios générés (réels)

— $\tilde{V}$ (resp. V) est la variance calculée sur les scénarios générés (réels)

— λ est un paramètre pour contrôler l'importance de la différence de variance dans l'entraînement du modèle

où la valeur de lambda est fixée à 2 car empiriquement, les modèles n'avaient pas de problème à reproduire un échantillon de moyenne semblable aux vraies données.

3.4 Méthodologie d'évaluation des résultats

Un enjeu du projet est d'évaluer de manière pertinente nos données générées. Deux approches sont sollicitées, une approche quantitative et une approche qualitative. Dans un premier temps, une première méthode est de calculer les moments des séries générées sur un grand nombre d'itération et de comparer ces estimations aux moments estimés de la série originale. On effectue ainsi cette opération pour la moyenne, la variance, la skewness et la kurtosis de la série.

Une seconde méthode est ensuite d'observer si pour n scénarios générés, la série originale reste proche des scénarios générés. Pour un point de la série originale donné, on calcule n

3. Itération complète du modèle sur l'ensemble des données d'entraînement

scénarios de log return et on trace l'évolution de la série sous différents scénarios. On compte ensuite le nombre de fois que la vraie série est sortie de l'intervalle créé par nos scénarios [4]. Formellement, on a alors :

$$\hat{\phi}(\hat{S}^{(1)}, ...\hat{S}^{(n)}, S) = \sum_{t=1}^{k} \mathbb{1}\{\min_{1\leq j\leq J} \hat{S}_t^{(j)} \geq r_t \cup \max_{1\leq j\leq J} \hat{S}_t^{(j)} \leq S_t\}$$

Où $\hat{S}_t^{(j)}$ [5] désigne l'observation t pour la série générée j et S_t l'observation t de la vraie série. On observe ensuite comment évolue le score lorsque l'on augmente le nombre de scénarios générés [6].

La troisième approche, visuelle, consiste à réduire la dimension des échantillons à l'aide d'un algorithme de réduction de dimension (dans notre cas le t-SNE [21]) afin de projeter graphiquement les données réelles et générés et d'observer si ces dernières sont proches.

Enfin, on testera graphiquement certains faits stylisés présentés dans la partie 4.3 et on vérifiera si ces caractéristiques sont conservés par les séries synthétiques.

4 Résultats

Dans cette section, on présente les résultats de trois modèles étudiés :

— Le modèle FF-GAN simple dont l'architecture est composé d'un réseau de neurone feed forward.

— Le modèle LSTM-GAN dont l'architecture fait intervenir un réseau Long Short Term Memory.

— Le modèle DC-GAN qui possède des couches de convolutions unidimensionnelles.

Une pénalité de gradient de type WGAN-GP [7] est appliquée aux deux derniers modèles. Afin d'éviter le surapprentissage, chaque modèle possède un paramètre de drop-out fixé à 0.4. Le drop-out est une méthode simple afin d'éviter le surrapprentissage des réseaux de neurones en mettant aléatoirement une partie des paramètres à 0 lors de l'apprentissage afin d'éviter que le modèle ne soit trop dépendant d'un petit nombre de paramètre [20].

4. On appellera cette intervalle la *fenêtre synthétique*

5. On définit $\hat{S}_t^{(j)} = S_0 * \prod_{i=1}^{t}(1 + \hat{r}_i^{(j)})$ où S_0 désigne l'observation de départ du scénario.

6. L'algorithme pour calculer l'évolution du score en fonction du nombre de scénarios générés se trouve en annexe 6

7. Wasserstein Gan Gradient penalty

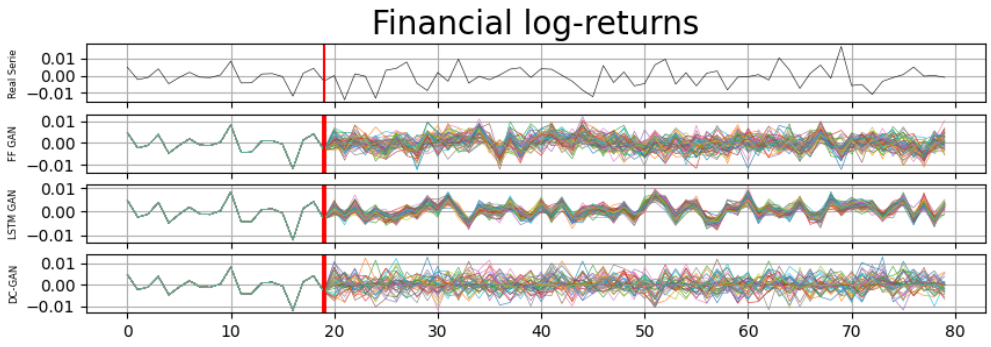


FIGURE 6 – Générations de 40 scénarios de log return par modèles

On observe dans un premier temps sur la figure 6 que l'amplitude des données générées par le modèle GAN simple semble respecter les vraies valeurs de la série. On remarque cependant un problème auquel on s'attendait, les 2 premiers modèles semblent générer des scénarios trop peu diversifiés. C'est particulièrement le cas pour le modèle LSTM GAN, dont tout les scénarios générés suivent une même trajectoire. La modification d'hyper-paramètres du modèle ne change pas sensiblement les résultats pour le modèle LSTM. Cette analyse visuelle n'est cependant pas suffisante, il convient maintenant de calculer pour chaque modèle si les séries générées possèdent des moments proches de la série réelle.

4.1 Estimation des moments

La table 1 présente la proximité des moments estimés des séries générées et de la série réelle. On remarque que la moyenne et la variance sont relativement bien reproduites par les séries générées. Cependant, la skewness et la kurtosis des données générées restent éloignées de la série réelle. On peut dans un premier temps penser que cela est dù au faible nombre de données d'entraînements. On remarque que sur nos 2494 données, moins de 5% des observations s'apparentent à des valeures extrêmes [8]. Le manque de données peut en partie expliquer la faible apparition de valeurs extrêmes dans les données générées.

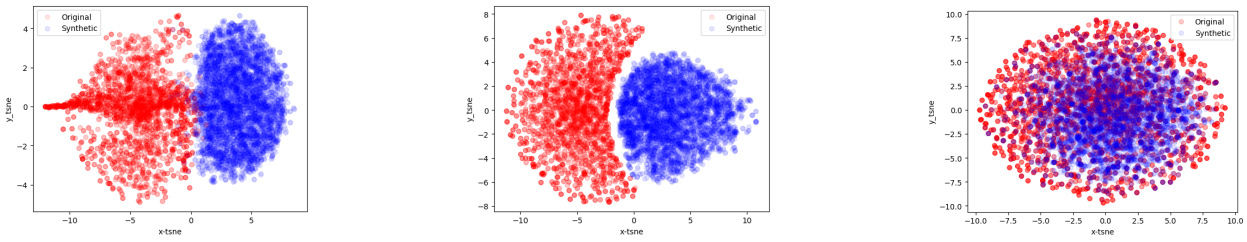
	Moyenne	Variance	Skewness	Kurtosis
FF GAN	0.000208	0.001713	0.358250	3.467750
LSTM GAN	0.000065	0.000405	0.362124	3.507560
DC-GAN	0.000215	0.000037	0.334039	2.295499

TABLE 1 – Différence en valeur absolue des moments estimés pour les modèles et la vraie série.

8. On entend ici une valeur qui se situe au dessus et en dessous des quantiles $q_{0.975}$, $q_{0.025}$ si la série suivait une distribution gaussienne.

4.2 Vraisemblance et diversité des données

Ensuite, on peut remarquer sur la figure 7 qu'en projetant les données sur un axe à deux dimensions, les données réelles et synthétiques sont facilement différenciables pour les modèles FF-Gan et LSTM-Gan. Cependant, pour le DC-Gan, on remarque que les données synthétiques et réelles semblent proches. Plus précisemment, les données synthétiques couvrent une majeure partie de la distribution des vraies données, mais les données plus extrêmes (aux extrémités du nuage de point) ne sont que faiblement représentées par les données synthétiques. Ce résultat rejoint ce qui avait été observé dans un premier temps, le modèle peine à générer des observations "rares", marquées par une forte volatilité. Les générations ne couvrent donc qu'une partie de la distribution réelle des données.



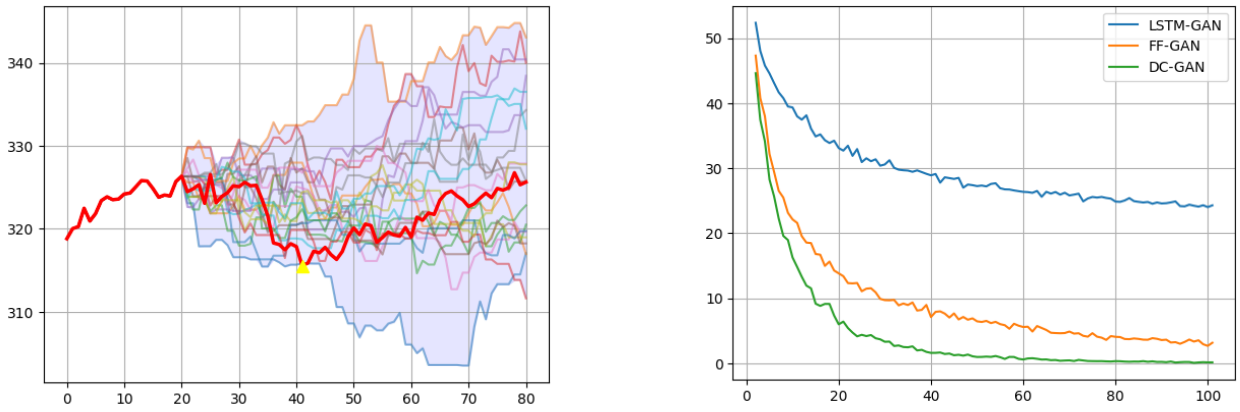
(a) FF-Gan (b) LSTM-Gan (c) DC-Gan

FIGURE 7 – Visualisation par t-SNE

Les données rouges représentent les échantillons de la série réelle et bleues les échantillons synthétiques

Pour évaluer la vraisemblance des données, bien que la tâche du projet ne soit pas de faire de la prévision, on peut observer le nombre moyen de données réelles en dehors de la fenêtre de données synthétiques. La figure 8a montre un exemple graphique de la métrique. On calcule donc le nombre de sortie de l'intervalle moyen pour n séries synthétiques générées. On obtient alors un graphique de la forme suivante 8b. On remarque que la très faible diversité des scénarios générées par le LSTM-GAN amène à un nombre très élevé d'observations en dehors de la fenêtre synthétique. Cette métrique permet de s'assurer de la diversité des scénarios générées et de leur proximité avec les données réelles. Cependant, il est important de considérer qu'un modèle générant des scénarios trop extrême aura très peu d'observations en dehors de la fenêtre des données synthétiques, mais cela ne signifie pas pour autant que les scénarios synthétiques sont réalistes.

Ces premiers résultats semblent plutôt favorable pour le modèle DC-GAN mais cette approche préliminaire n'est pas suffisante. Il convient maintenant de regarder quelques faits stylisés que respectent nos vraies séries financières et de les comparer aux données synthétiques. Une partie des résultats dans la prochaine section découleront du modèle entraîné sur l'index S&P 500 afin de pouvoir bénéficier d'un plus grand nombre d'observations (23000 observations environ) et de vérifier certains faits stylisés propres aux indices financiers.



(a) Génération de 20 scénarios de prix.

(b) $\hat{\phi}$ en fonction de j

FIGURE 8 – Comparaison de deux graphiques

La ligne rouge représente un échantillon réel. Le triangle jaune signifie qu'une observation est sortie de la fenêtre des données synthétiques (zone hachurée).

4.3 Comparaison des caractéristiques des séries financières

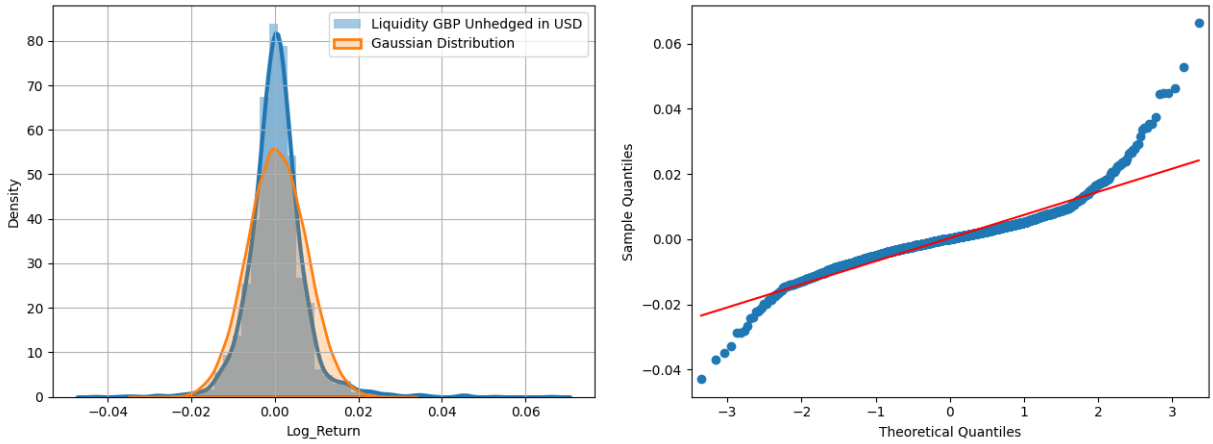
Tout d'abord, il a été remarqué que les valeurs générées par le modèle DC-GAN étaient rarement des valeurs extrêmes. En traçant un grand nombre de scénarios, on remarque qu'on observe pas réellement de valeurs extrêmes. Si dans la série originale, les rendements les plus hauts atteignent 0.15 en valeur absolue, aucune génération dans nos données n'a dépassé 0.06. On remarque donc naturellement que les queues de distribution de nos séries générées sont moins épaisses que les données réelles. Ainsi le but de cette dernière section est de passer en revue les faits stylisés sur les données réelles et générées pour avoir une description statistique de nos données et voir l'efficacité des modèles.

Nous nous intéréssons à la stationnarité des rendements et nous effectuerons pour cela d'abord un test [9] de racine unitaire Augmented Dickey-Fuller2 dont les résultats sont présentés dans le tableau 3. On rejette l'hypothèse nulle de présence de racine unitaire pour la série des rendements qui nous montre qu'elle est bien stationnaire. Cela est respecté dans nos données réelles et dans les séries générées également. Cela montre que le modèle ne peine pas trop à retrouver cette caractéristique.

Ensuite, en s'intéressant à la non-normalité, nous observons ce fait dans la figure 9 qui nous montre un qqplot qui s'éloigne au niveau des queues. Du côté des séries générées, nous présentons la figure 10 qui montre que globalement la notion de queue épaisse est respectée mais que le centre de la distribution est mal fitée [10].

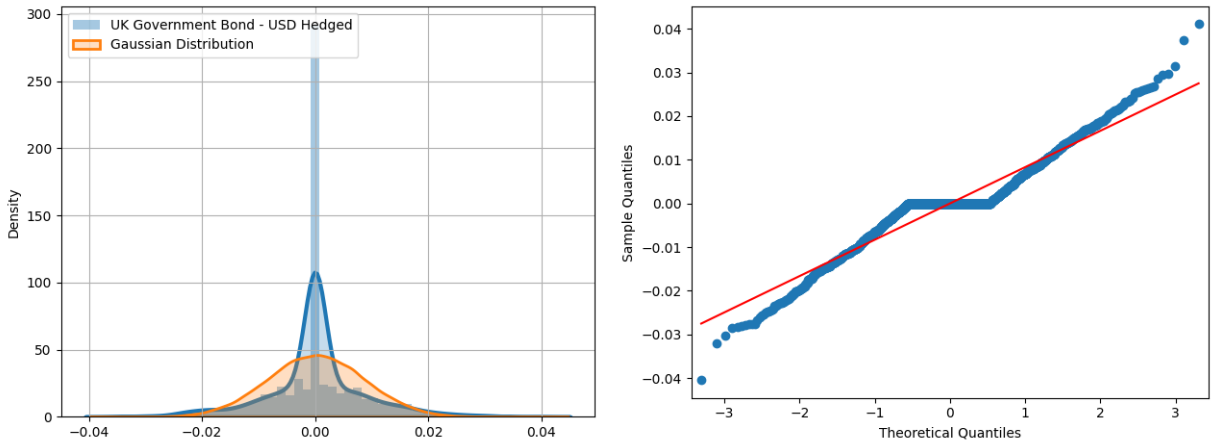
9. Tous les tests sont décris en annexe

10. Le pic très important à 0 est dû à un grand nombre de log return générés à 0. Ce fait est une conséquence du paramètre de drop-out fixé à une valeur élevée qui est un prix à payer pour limiter l'overfitting. Cela reste cependant un bon exemple de "No free lunch theorem" en Machine Learning.



(a) Distribution (b) QQPlot

FIGURE 9 – Comparaison de la distribution de l'actif UK Government Bond - USD Hedged avec la loi normale (vraies données)



(a) Distribution (b) QQPlot

FIGURE 10 – Comparaison de la distribution de l'actif UK Government Bond - USD Hedged avec la loi normale (données générées).

Nous fournissons également les résultats du test de Jarque-Bera qui rejettent l'hypothèse nulle de normalité des rendements pour les deux séries sont présents dans la table 2. Néanmoins, on voit en observant la valeur de la statistique de test qui est largement plus élévée pour les données réelles, que la distribution des données synthétiques n'est probablement pas autant leptokurtique que les données réelles.

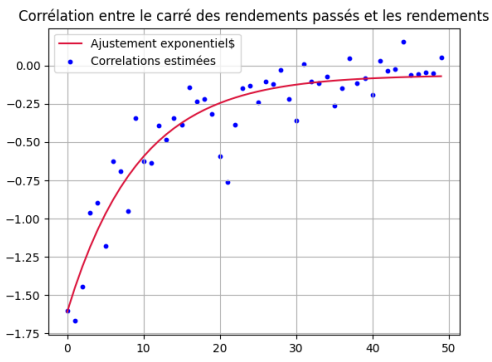
	Jarque-Bera	
	Réelles	Synthétiques
Statistique	10964	485
p-value	< 0.01	< 0.01

TABLE 2 – Tests de normalité des données réelles et synthétiques (DC-GAN)

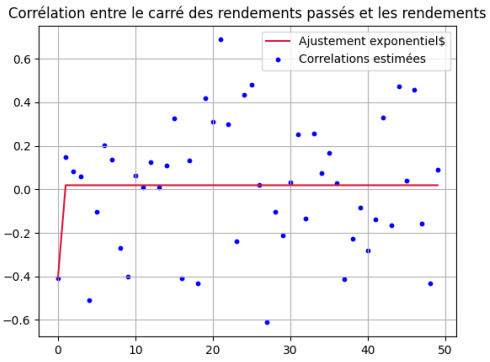
	ADF		ADF(constante)		ADF (constante et trend)	
	Réelles	Synthétiques	Réelles	Synthétiques	Réelles	Synthétiques
Statistique	-8.415	-9.011	-8.527	-9.034	-14.631	-9.031
p-value	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01

TABLE 3 – Tests de stationnarité ADF des données réelles et synthétiques(DC-GAN)

Ensuite, nous regardons le leverage effect. Les détails de ce fait se trouvent en annexe 6. Graphiquement, la figure 11a montre que les autocorrélations sont négatives puis tendent vers 0 pour le S&P 500, tandis que ce fait n'est pas vérifié pour les données synthétiques. On observe des autocorrélations qui ne suivent pas de schéma particulier pour nos données générées. On note que l'effet de levier ne concerne que les indices et les actions et que sur notre série des bonds du gouvernement UK, ce fait ne pouvait pas être vérifié.



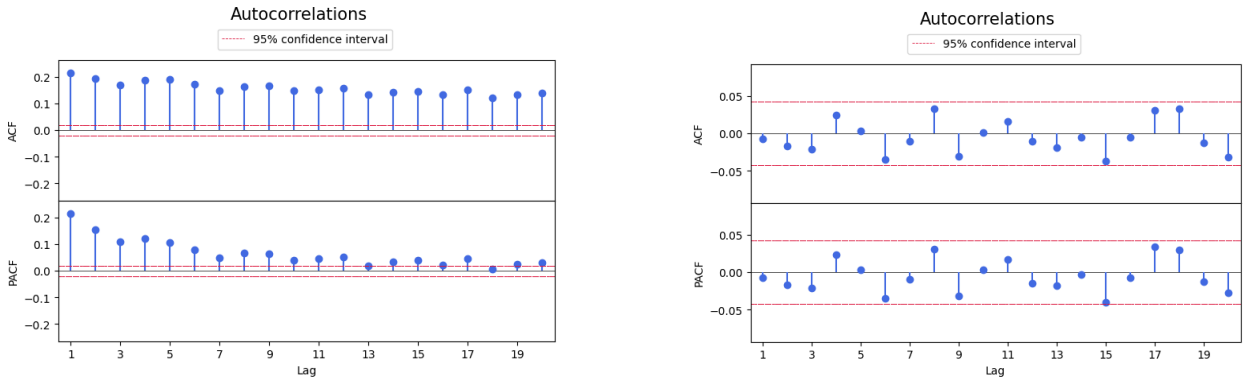
(a) S&P 500.



(b) Données synthétiques

FIGURE 11 – Représentation graphique du leverage effect.

Enfin, on s'intéresse au phénomène de clustering de volatilité. On se penche particulièrement sur l'autocorrélation des rendements en valeurs absolues avec les valeurs passées. Cet effet traduit des chocs qui se propagent, comme le modélisent les modèles de type GARCH. La figure 12a montre que les chocs sont bien influencés par les chocs passés et que cette autocorrélation perdure dans le temps. D'un autre côté, la figure 12b montre que ce phénomène n'est pas observé pour nos séries synthétiques.



(a) SP 500.

(b) Données synthétiques

FIGURE 12 – Représentation graphique du clustering de volatilité.

Ces résultats dévoilent un point de vue nuancé sur nos données synthétiques. La plupart de ces faits n'ont pas su être reproduits par le modèle malgré un nombre de données d'entraînement élevé.

5 Conclusions et discussions

La génération de données est un enjeu important dans le domaine de la finance et de l'assurance. L'utilisation de modèles stochastiques telles que Black & Scholes [6] ou encore Heston [14] a pour avantage de reposer sur un cadre théorique solide et un fonctionnement du modèle bien connu. D'un autre côté, les méthodes émergentes de machine learning tiennent leur légitimité sur leurs fondements théoriques mais sur leurs résultats empiriques. Notre travail a consisté à adapter des architectures de GAN bien connues pour les appliquer à des séries financières. Nos résultats montrent que certaines architectures comme les LSTM ne parviennent pas à s'imprégner de la distribution des données financières. D'un autre côté, l'utilisation de réseaux de neurones à convolution semble plus concluante et parvient à générer des données diversifiées. Cependant, nos modèles GAN n'ont pas réussi à générer des valeurs extrêmes, telles que celles observées dans la série réelle. De plus, certaines caractéristiques profondes inhérentes aux données financières ne sont pas reproduites par les données synthétiques, même avec un grand nombre de données d'entraînement

Les modèles GAN s'entraînent à l'aide d'un discriminateur qui guide le générateur dans son apprentissage. Il est donc nécessaire que le générateur et le discriminateur aient une structure adéquate afin de prendre en compte tous les aspects du problème. Ici, nous avons constaté que les modèles GAN "simples" n'ont pas été suffisants pour capturer toutes les caractéristiques des séries financières. Il a cependant été démontré que l'architecture du modèle avait un fort impact sur l'efficacité du modèle à créer des données vraisemblables.

Les GAN ont un grand potentiel pour répondre aux besoins spécifiques de données financières synthétiques, ainsi qu'à d'autres tâches de modélisation diverses. Nous avons remarqué que ces modèles pouvaient s'approprier certaines caractéristiques de la distribution sans nécessiter de poser des hypothèses particulières ou de guider le modèle, mais simplement à l'aide d'une série financière.

Néanmoins, on observe qu'ils sont plus présents dans la recherche que dans la pratique, mais leur potentiel est à explorer. Les challenges qui sont sous-jacents tournent également autour de l'évaluation de la qualité et de la fiabilité des prédictions qui n'est pas toujours évident en raisons des faiblesses citées auparavant. Les limites que nous retrouvons notamment dans le respect des faits stylisés n'est pas quelque chose que nous retrouvons uniquement sur ces modèles mais également dans les modèles stochastiques plus classiques qui ont du mal parfois à les retrouver à cause d'hypothèse très fortes. Ainsi, cela ne doit pas forcément être interprété comme une faiblesse des modèles GAN mais plutôt comme une piste de recherche à approfondir.

D'autre part, nous avons pu observer qu'un essai de différentes structures et architectures nous a permis d'atteindre des résultats très différents. Ainsi, combiner ces différentes approches peut être un point clé intéréssant pour les futures recherches et améliorer les résultats ou même toucher d'autres aspects plus complets comme notamment la génération de séries multivariées en prenant en compte les relations entre les différentes séries.

Références

[1] Soumith Chintala Alec Radford, Luke Metz. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[4] Josef ARLT and Markéta ARLTOVÁ. Financial time series and their.

[5] Md Abul Bashar and Richi Nayak. Algan : Time series anomaly detection with adjusted-lstm gan, 2023.

[6] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of political economy*, 81(3) :637–654, 1973.

[7] Jean-Philippe Bouchaud, Andrew Matacz, and Marc Potters. Leverage effect in financial markets : The retarded volatility model. *Physical review letters*, 87(22) :228701, 2001.

[8] Anirban Chakraborti, Ioane Toke, Marco Patriarca, and Frédéric Abergel. Econophysics review : I. empirical facts. *Quantitative Finance*, 11 :991–1012, 07 2011.

[9] Rama Cont. Empirical properties of asset returns : stylized facts and statistical issues. *Quantitative finance*, 1(2) :223, 2001.

[10] Florian Eckerli and Joerg Osterrieder. Generative adversarial networks in finance : an overview. *arXiv preprint arXiv :2106.06364*, 2021.

[11] Paul Glasserman. *Monte Carlo methods in financial engineering*, volume 53. Springer, 2004.

[12] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[13] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

[14] Steven L. Heston. A closed form solution for options with stochastic volatility with application to bonds and currency options. *The Review of Financial Studies*, 6(2) :327–343, 1993.

[15] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8) :1735–1780, 1997.

[16] Stefan Jansen. *Machine Learning for Algorithmic Trading : Predictive models to extract signals from market and alternative data for systematic trading strategies with Python*. Packt Publishing Ltd, 2020.

[17] Gautier Marti. Corrgan : Sampling realistic financial correlation matrices using generative adversarial networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8459–8463. IEEE, 2020.

[18] Gautier Marti, Victor Goubet, and Frank Nielsen. ccorrgan : Conditional correlation gan for learning empirical conditional distributions in the elliptope. In *International Conference on Geometric Science of Information*, pages 613–620. Springer, 2021.

[19] Lu Mi, Macheng Shen, and Jingzhao Zhang. A probe towards understanding GAN and VAE models. *CoRR*, abs/1812.05676, 2018.

[20] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout : A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56) :1929–1958, 2014.

[21] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9 :2579–2605, 2008.

[22] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86) :2579–2605, 2008.

[23] Christian Bakke Vennerød, Adrian Kjærran, and Erling Stray Bugge. Long short-term memory rnn, 2021.

[24] Guangxuan Zhu, Hongbo Zhao, Haoqiang Liu, and Hua Sun. A novel lstm-gan algorithm for time series anomaly detection. In *2019 prognostics and system health management conference (PHM-Qingdao)*, pages 1–6. IEEE, 2019.

6 Annexes

Annexe 1 : Deep Convolutional Generative Adversarial Network (DCGAN)

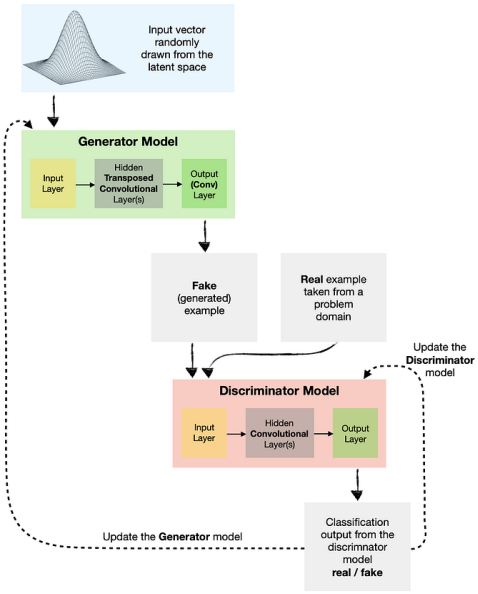


FIGURE 13 – Deep Convolutional Generative Adversarial Network (DCGAN). Source : Towards Data Science

Annexe 2 : Divergence JS et distance de Wasserstein

En utilisant les mêmes notations que dans le papier, la divergence JS est définie par :

$$D_{JS}(P_{data}\|P_g) = D_{KL}(P_{data}\|P_m) + D_{KL}(P_g\|P_m)$$

où $P_m = \frac{P_{data}+P_g}{2}$

Définissons formellement la distance de Wasserstein :
Soit B une métrique compacte et $Prob(B)$ l'espace de toutes les mesures de probabilités définies sur B. On note deux distributions $P_r, P_s \in Prob(B)$ alors la distance de Wasserstein est définie par :

$$W(P_r, P_s) = \inf_{\gamma \in \Pi(P_r, P_s)} \mathbb{E}_{(x,y)\sim\gamma(x,y)}\|x - y\|$$

où $\Pi(P_r, P_s)$ est l'ensemble de toutes les distributions jointes $\gamma(x, y)$ qui ont pour distributions marginales respectivement P_r et P_s.

Annexe 3 : Fonction k-Lipschitzienne

Soient (E, d_E) et (F, d_F) des espaces métriques, $f : E \to F$ une application et k un réel positif.

On dit que f est k-Lipschitzienne si :

$$\forall (x, y) \in E^2, \quad d_F(f(x), f(y)) \leq k \cdot d_E(x, y).$$

où d_E et d_F sont deux distances sur E et F respectivement.

Annexe 4 : Algorithme d'évaluation des scénarios

Algorithm 2 Algorithme calcul de la couverture des scénarios synthétiques

for $j = 2$ à J **do**
    for $i = 1$ à n **do**
        start = random(0 :T) (taille maximale de la série)
        Générer j scénarios de log returns $\hat{r}^{(1)}...\hat{r}^{(j)}$
        Pour chaque scénario, calculer l'évolution de la série synthétique au point de départ S_{start}
        Calculer $\hat{\phi}(\hat{S}^{(1)}, ...\hat{S}^{(j)}, S)_i^{(j)}$
    end for
    $\overline{\phi}^{(j)} = \frac{1}{n} \sum_{t=1}^{n} \hat{\phi}_i^{(j)}$
end for
Return $[\overline{\phi}^{(2)}, \overline{\phi}^{(3)}...\overline{\phi}^{(J)}]$

Annexe 5 : Test de Jarque Bera

Introduit en 1980 par Carlos Jarque Uribe et Anil K. Bera, le test de Jarque-Bera est test d'hypothèse non paramétrique permettant de tester si une variable continue suit une loi normale.

La particularité du test de Jarque-Bera repose sur le fait qu'il n'étudie pas directement l'adéquation à la loi normale mais plutôt simultanément l'égalité du coefficient d'asymétrie (skewness) et celui d'aplatissement (appelé également coefficient of kurtosis) à ceux attendus dans le cadre d'une loi normale.

On teste donc H_0 : les données suivent une loi normale contre H_1 : les données ne suivent pas une loi normale.

La statistique de test est donnée par$JB = \frac{n}{6}\left(\beta_1^2 + \frac{(\beta_2-3)^2}{4}\right)$ où n est le nombre d'observations, β_1 le coefficient d'asymétrie de l'échantillon testé et enfin β_2 la kurtosis de l'échantillon testé.

$$S = \frac{\mu_3}{\sigma^3} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^3}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^2\right)^{3/2}} \tag{2}$$

$$K = \frac{\mu_4}{\sigma^4} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^4}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^2\right)^{2}}, \tag{3}$$

où μ_3 et μ_4 sont les moments d'ordre 3 et 4 des données. La statistique JB suit asymptotiquement une loi du χ^2 à deux degrés de liberté.

Annexe 6 : Dickey-Fuller Augmenté (ADF)

Soit un processus $\{X_t\}_T$. On dit que $\{X_t\}$ est stationnaire au sens faible si :

1. $\forall t \in \mathbb{Z}, E(X_t) = \mu \in \mathbb{R}$
2. $\forall t \in \mathbb{Z}, V(X_t) = \sigma_x^2 < +\infty$
3. $\forall t, h \in \mathbb{Z}, Cov(X_t, X_{t-h}) = \gamma(h)$

Dans la stratégie séquentielle du test ADF, on considère 3 modèles :

$$X_t = \rho X_{t-1} + \sum_{i=1}^{p} \Delta X_{t-i} + u_t \tag{M1}$$

$$X_t = \alpha + \rho X_{t-1} + \sum_{i=1}^{p} \Delta X_{t-i} + u_t \tag{M2}$$

$$X_t = \alpha + \beta t + \rho X_{t-1} + \sum_{i=1}^{p} \Delta X_{t-i} + u_t \tag{M3}$$

Les hypothèses du test ADF :

$$\begin{cases} H_0 : |\rho| = 1 \\ H_1 : |\rho| < 1 \end{cases}$$

Sous H_0, la statistique de test est donnée par :

$$t_\rho = \frac{\hat{\rho} - 1}{\hat{\sigma}_{\hat{\rho}}}$$

Sous condition que $X_0 = 0$ (en pratique on utilise la série $Y_t = X_{t+1} - X_t$ pour se passer de cette condition), on va rejeter H_0 si $t_\rho < \mathcal{T}$, où $\mathcal{T}$ est la valeur critique de la statistique sous H_0. Les valeurs critiques de la statistique sont données par la table de Dickey-Fulleren fonction des modèles(sans constante et sans trend ; avec constante et avec constante et trend). Si on rejette l'hypothèse nulle H_0 alors la série est stationnaire.

Annexe 7 : Leverage effect

L'effet de levier montre une corrélation négative entre la volatilité et les rendements d'un actif. Cette effet levier est observé empiriquement pour les indices et les actions. Formellement on a :

1. $X_t = ln(S_t)$ où S_t représente le prix de l'actif en t.
2. $\langle X \rangle_{s,t} = \lim_{\|\pi\| \to 0} \sum_{i=0}^{n-1} (X_{t_{i+1}} - X_{t_i})^2$ où $\pi = t_0, t_1 ... t_n$ est une grille sur $[s, t]$ et $\|\pi\|$ est la taille maximale tous les sous-intervalles de π
3. $\langle X \rangle_{s,t}$ représente la volatilité théorique entre X_t et X_s

L'effet de levier se traduis par : $\frac{E[\langle X \rangle_{t,t+1} X_1]}{E[\langle X \rangle_{t,t+1}]^2} = -Ae^{-\frac{t}{L}}$ où $A > 0$ et L est l'échelle de corrélation. Source. Graphiquement, on attends donc que l'estimation des corrélations soit négative et tends vers 0 lorsque t augmente.

Annexe 8 : Générations de log return (UK Government Bond), DC-GAN

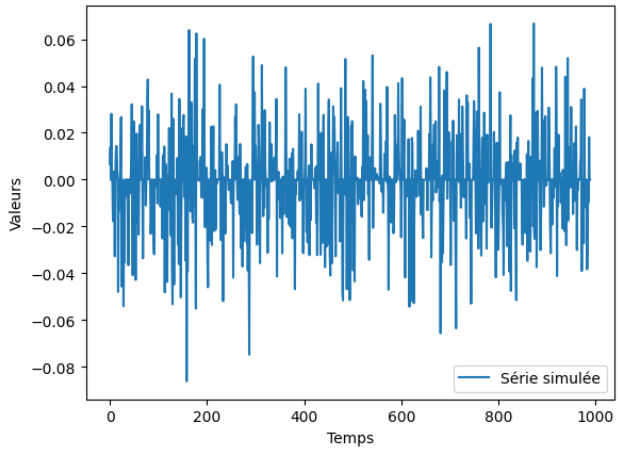
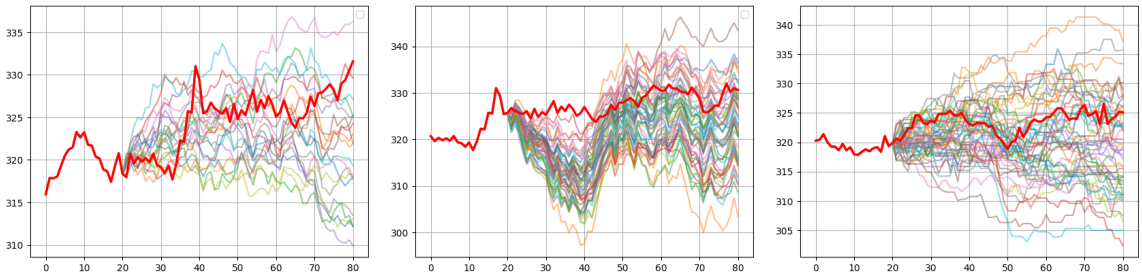


FIGURE 14 – Log return UK Government Bond généré, DC-GAN

Annexe 9 : Génération de scénarios de prix par modèle GAN



(a) FF-GAN (b) LSTM-GAN (c) DC-GAN

FIGURE 15 – La ligne rouge représente une observation réelle de la série UK Government Bond

Annexe 10 : Simulation à long terme

Les simulations ci-dessous sont issues du modèle DC-GAN, entraîné sur la série UK Government bond. La ligne bleue représente la vraie trajectoire de la série. On remarque que sur une simulation de long terme (plus de 1500 log returns générés), certains scénarios restent proches de la série originale.

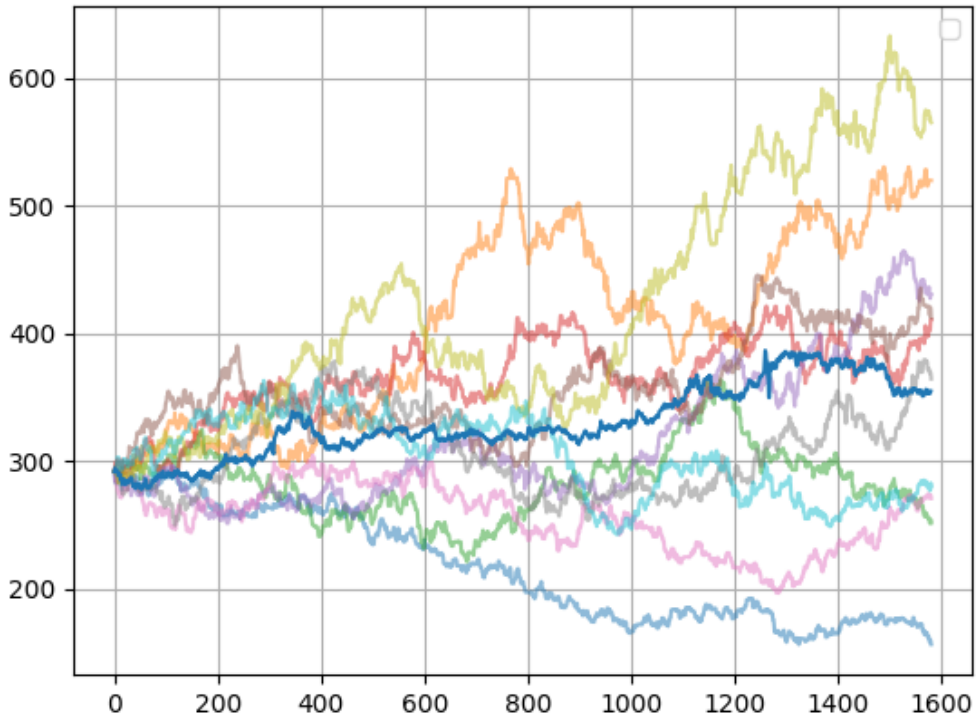


FIGURE 16 – Générations de scénarios de long terme, DC-GAN