AN EULER-BASED GAN FOR TIME SERIES

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce a new model of generative adversarial network (GAN) for time series generation based on Euler embedding together with Wasserstein metric including Sinkhorn divergence. We observe that the Euler embedding improves the stability of learning, provides meaningful learning parameters such as drift and volatility while allowing the representation of a large class of underlying time series. We demonstrate the capacity of the Euler GAN simulator to replicate classical Monte Carlo simulations, even in a possibly large multi-dimensional setting. The generator can retrieve correlation structure or independence property. Combining the proposed methodology with transfer learning methods, we verify that our generator can learn efficiently from one single time serie trajectory. The approach is illustrated on S&P500 trajectories.

1 INTRODUCTION

Monte Carlo simulations of time series are widely used for industrial applications (stress testing, investment decisions, stochastic control, weather forecast, anomaly predictions...). They are in particular extensively used in the financial sector, for market stress testing (Sorge (2004)), risk control and deep hedging (Longstaff & Schwartz (2001); Buehler et al. (2019); Fécamp et al. (2019)) or for the measurement of common risk indicators such as Value at Risks (Jorion (2000)) among others. Providing accurate Monte Carlo simulations of time series is a challenging task, which requires underlying modelling assumptions on the time dependency of the variables. The design of a realistic and tractable model remains a tedious and mainly manual task. Monte Carlo simulators are required to select one type of models, such as $AR(p), ARMA$, Black-Scholes or Heston model, which are very common references in the industry. Hereby, it is not straightforward to update these models when data of a new kind is observed (e.g. apparition of negative interest rates or negative prices in electricity markets, an economic crisis modifying the market structure or new meteorological conditions). Updating the model in these situations is long and costly. This naturally calls for the development of reliable model free data driven generators for time series.
Generative methods such as Variational Auto Encoders (VAE) and Generative Adversarial Networks (GAN) show impressive results for text, images or video data. Developing similar generative methods for time series application is very appealing as it directly learns a reliable diffusion model from the data, without underlying modelling assumption. However, due to the complex and possibly non stationary underlying time structure of the initial time series, such generative methods are very difficult to apply as such. Figure 1 illustrates one of the difficulty: the generator is able to learn properly the marginal distribution of the time series at each date, whereas it is not able to capture the underlying dynamics in time of the process.

The bootstrap method proposed by (Efron (1982)) is one of the first data-driven model free attempt to generate time series data directly from historical dataset. Random samples of the data are simply taken randomly with replacement. The scope of this technique is limited as it does not generate additional synthetic data but only historical ones. A first attempt towards GAN application to time series has been recently proposed in Yoon et al. (2019), followed by Yu et al. (2017) or Luo et al. (2018). More recently, some literature focused on the embedding of time series and stochastic processes using signature Fermanian (2019) or Fourier Wasserstein distance Steinerberger (2018). A specific application if GANs to commodity markets is detailed in Schreiber et al. (2019), as well as in Fu et al. (2019). Several papers combine transport optimal theory with generative methods in order to improve distribution learning Arjovsky et al. (2017), Genevay et al. (2018), while Xu et al. (2020) focus on sequence generation. In these contributions, the required volume of training data is

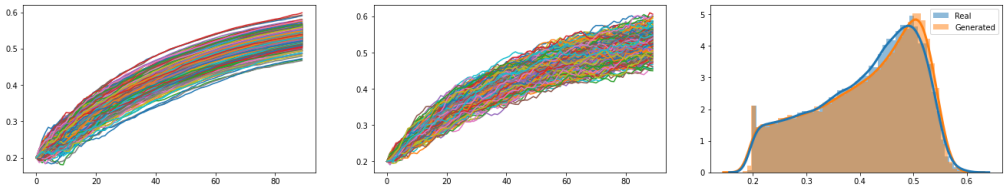

Figure 1: **Left:** A generated time series which marginal at each date is equal to the target time series of the **Middle**. **Right:** Distribution of the time-flattened output vectors for the generator and of the real sample

huge, and stationarity properties of the time series are also usually required.
In this paper, we build generators based on a Euler embedding of the time series, focusing hereby on the general temporal structure of Ito processes. This formulation enables a rigorous mathematical formulation of the problem as well as considering relevant constraints on the generator. Such embedding can also be generalized by adding jumps or autocorrelation structures. We develop three Euler based GANs for time series, relying on several distances between marginal distributions, including 1-Wasserstein, and Sinkhorn distance. We verify that the GANs can learn to replicate a Monte Carlo simulator of one or multi-dimensional classical stochastic processes. It recovers the underlying correlation or independence structure and can scale up to dimension 20 in our experiments. We then perform a transfer learning approach on the S&P 500 time series, allowing to train our method on a simulator and then to fine tune the model on the real data points. Mixing properly real data and synthetically generated Monte Carlo simulations, we observe that the GAN can properly learn time series from a single trajectory observation and provide relevant risk indicators such as Value at Risk in such framework.

Main Contributions:

- Combining an Euler embedding with a GAN structure, we exhibit a reliable time series generator, learning the underlying trend and volatility structures. The process space spanned by the Euler embedding is large enough for most applications, and can easily be extended to include auto regressive time series or series with jumps.
- Using different distances between distributions (Jensen-Shannon, Wasserstein, Sinkhorn), we are able to take into account the geometry of multivariate sequence space, allowing to capture non-linear temporal dynamics.
- We propose an innovative way to combine usual Monte Carlo together with GANs for risk management purpose, improving liability and risk criterion for financial applications (financial derivative pricing and Value at Risk computation).
- We develop a reliable transfer learning approach when few data points are available, by first learning on synthetic Monte Carlo based data points.
- Our generators capture correlation and independence between time series and scale up to dimension 20 in our experiments.

2 Problem description and Euler GAN design

We are given samples of a time series $X \in \mathbb{R}^d$ starting from a same point X_0 and observed on the time grid $\pi = t_0 < t_1 < ... < t_N$. X may be considered as a random vector defined on $\mathbb{R}^N \times \mathbb{R}^d$. We assume we do not know the distribution $\mathbb{P}$ of X. Our objective is to find a generator g_θ that generates a random vector $\hat{X} \in \mathbb{R}^N \times \mathbb{R}^d$ having distribution $\mathbb{P}^\theta$ the closest possible to $\mathbb{P}$. The time series X considered here is not supposed to be stationary, in comparison to the current literature on the topic. For example, the log return of a Black-Scholes diffusion provides stationary gaussian iid samples, and considerably ease our problem.

2.1 SDE REPRESENTATION OF THE GENERATOR

In our GAN setting, the generator (g_θ) is in charge of producing $\hat{X}^\theta$ through two measurable functions b^θ, σ^θ representing respectively the drift and the volatility, as well as a covariance matrix Γ^θ, output of a neural network parameterized by θ. The generated continuous time process $\hat{X}^\theta$ is by construction following the stochastic differential equation:

$$d\hat{X}_t^\theta \quad = \quad b^\theta(t, \hat{X}_t^\theta)dt + \sigma^\theta(t, \hat{X}_t^\theta)dB_t. \tag{1}$$

with B a d-dimensional Brownian motion on some probability space $(\Omega, \mathcal{F}, \mathcal{P})$ where $(\mathcal{F}_t)_{t_0 \leq t \leq t_N}$ is a filtration satisfying the usual conditions. This representation gives a temporal structure to $\hat{X}^\theta$ and differs from the usual applications of GANs and VAE to time series. For given b^θ, σ^θ, generated samples of $\hat{X}^\theta$ are given by discretizing Equation (1) with an Euler embedding:

$$\begin{aligned} \hat{X}_0^\theta &= X_0 \\ \hat{X}_{t_{i+1}}^\theta &= \hat{X}_{t_i}^\theta + b^\theta(t_i, \hat{X}_{t_i})(t_{i+1} - t_i) + \sigma^\theta(t_i, \hat{X}_{t_i})(t_{i+1} - t_i)Z. \end{aligned} \tag{2}$$

where $Z \sim N(0, \Gamma^\theta)$. During the training phase, g_θ calibrates two functions b^θ and σ^θ and builds a process trajectory $\hat{X}^\theta$ following Euler scheme (2) from a noise vector Z, then a discriminator d_φ (parameterized by $\varphi \in \theta$) tries to distinguish between fake times series generated by the Euler GAN and real time series. A control of drift and volatility is possible with generator last layers activation function, for example with a tanh to force drift in $[-1, 1]$ interval. For a continuous distribution $\mathbb{Q}$ and samples x drawn from $\mathbb{Q}$ we denote by $\Pi_d(x)$ the empirical distribution obtained from x.
In order to emphasize the robustness of our approach, we investigate three different GAN representations of for time series:

- **Jensen–Shannon divergence:** As defined in Goodfellow et al. (2014), we intend to solve

$$\inf_\theta \sup_\varphi \mathbb{E}_{X \sim \mathbb{P}}[\log(d_\varphi(X))] + \mathbb{E}_{X^\theta \sim \mathbb{P}^\theta}[\log(1 - d_\varphi(\hat{X}^\theta)]. \tag{3}$$

  Given samples x and $\hat{x}^\theta$ drawn respectively from $\mathbb{P}$ and $\mathbb{P}^\theta$, the term in the $\inf\sup$ may be estimated by:

$$l^{js}(\theta, \varphi, x, \hat{x}^\theta) \quad = \quad \mathbb{E}[\log(d_\varphi(x))] + \mathbb{E}[\log(1 - d_\varphi(\hat{x}^\theta)]. \tag{4}$$

  This setting, referred to as Euler GAN (eGAN) in this paper and its algorithm is described in Annex 2. Such approach is recognized to fail providing meaningful representations of disjoint distributions, see Arjovsky et al. (2017), and we investigate several alternatives.

- **1-Wasserstein distance:** One way to improve the training of vanilla GANs is to use a different distance between distributions. First, we replace the Jensen-Shannon divergence by the 1-Wasserstein metric, due to its nicer regularity properties. Wasserstein distance is for example intensively used in Optimal Transport field (see Villani (2008)). For a cost function $c : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$, the p-Wasserstein distance between two distributions $\mathbb{P}$ and $\mathbb{P}^\theta$ (defined on M) is given by:

$$\mathcal{W}_{c,p}(\mathbb{P}, \mathbb{P}^\theta) = \left(\inf_{\pi \in \Pi(\mathbb{P}, \mathbb{P}^\theta)} \int_{M \times M} c(x, y)^p d\pi(x, y) \right)^{\frac{1}{p}} \tag{5}$$

  As pointed out in Arjovsky et al. (2017), under the assumption that d_φ is K-Lipshitz and for $p = 1$, $c(x, y) = \|x - y\|_1$, Kantorovich-Rubinstein duality applies and Equation (5) is equal to:

$$\frac{1}{K} \inf_\theta \sup_{\varphi, ||d_\varphi||_L \leq K} \mathbb{E}_{X \sim \mathbb{P}}\left[d_\varphi(X)\right] - \mathbb{E}_{X^\theta \sim \mathbb{P}_\theta}\left[d_\varphi(X^\theta)\right] \tag{6}$$

  Given samples x and $\hat{x}^\theta$ respectively drawn from $\mathbb{P}$ and $\mathbb{P}^\theta$ the term inside the $\inf\sup$ in (equation 6) may be approximated by:

$$l^{W_1}(\theta, \varphi, x, \hat{x}^\theta) = \mathbb{E}\left[d_\varphi(x)\right] - \mathbb{E}_{X^\theta \sim \mathbb{P}_\theta}\left[d_\varphi(\hat{x}^\theta)\right]. \tag{7}$$

  This setting is referred to as eWGAN, the algorithm is described in Annex 3.

- **Regularized 2-Wasserstein distance:** It defines for $\lambda \geq 0$ by

$$\begin{aligned}\mathcal{W}_{\lambda,c}(\mathbb{P},\mathbb{P}_\theta)^p &= \inf_{\pi\in\Pi(\mathbb{P},\mathbb{P}^\theta)} \int_{M_1\times M_2} c(x,y)^p d\pi(x,y) \\ &+ \lambda \int_{M_1\times M_2} log\left(\frac{\pi(x,y)}{d\mathbb{P}(x)d\mathbb{P}_\theta(y)}\right) d\pi(x,y). \end{aligned} \tag{8}$$

As mentioned in Cuturi et al. (2019) the regularization term ensures that $\mathcal{W}_{\lambda,c}(\mathbb{P},\mathbb{P}_\theta)$ is θ-differentiable as well as computationally tractable using using the Sinkhorn algorithm. Let $c_\varphi(x,y) := \|d_\varphi(x) - d_\varphi(y)\|_2$. In order to reduce the bias in measuring the distance between the two distributions without reducing the regularization parameter λ, Genevay et al. (2018) proposes to use the following Sinkorn divergence:

$$SK(\mathbb{P},\mathbb{P}^\theta) = \inf_\theta \sup_\varphi 2\mathcal{W}_{\lambda,c_\varphi}(\mathbb{P},\mathbb{P}^\theta)^p - \mathcal{W}_{\lambda,c_\varphi}(\mathbb{P},\mathbb{P})^p - \mathcal{W}_{\lambda,c_\varphi}(\mathbb{P}^\theta,\mathbb{P}^\theta)^p \tag{9}$$

For two samples x, x^θ, let $W_{\lambda,c_\varphi}(x,x^\theta) = \mathcal{W}_{\lambda,c_\varphi}(\Pi_d(x),\Pi_d(x^\theta))$. The term inside the $\inf\sup$ may be approximated by:

$$l^{SK}(\theta,\varphi,x,x^\theta) = 2W_{\lambda,c_\varphi}(x,\hat{x}^\theta)^p - W_{\lambda,c}(x,x)^p - W_{\lambda,c}(\hat{x}^\theta,\hat{x}^\theta)^p \tag{10}$$

This setting is referred to as eSGAN and its algorithm is described in Algorithm 1.

In the two first configurations, the generator minimizes only the expectation of generated data (the second term in Equation 3 and 6) while the discriminator maximizes the whole loss. However, in Sinkhorn GAN, the generator minimizes the whole loss ℓ^{SK}.

Input: θ_0, φ_0 randomly chosen, α, β learning rates, K number of iterations,
M batch size, n_c critic iterations, c clipping value, $\left(X^{(i)}\right)_{i=1..M}$ real data
Output: θ, φ

$\theta \leftarrow \theta_0, \varphi \leftarrow \varphi_0$;
for $k = 1..K$ **do**
  for $j = 1..n_{critic}$ **do**
    $x \leftarrow M$ samples with $X^{(i)} = (X^{(i)}_{t_1}, \ldots, X^{(i)}_{t_N})_{i=1..M}$;
    $z \leftarrow M$ samples iid gaussian noise;
    $\hat{x}^\theta \leftarrow M$ generation from Euler scheme and $g_\theta(z)$;
    $\varphi \leftarrow \varphi + \alpha\text{Adam}\left(\nabla_\varphi(\ell^{SK}_{\lambda,c}(d_\varphi(x), d_\varphi(\hat{x}^\theta), \alpha\right)$;
    $\varphi \leftarrow \text{clip}(\varphi, -c, c)$;
  end
  $x \leftarrow M$ samples with $X^{(i)} = (X^{(i)}_{t_1}, \ldots, X^{(i)}_{t_N})_{i=1..M}$;
  $z \leftarrow M$ samples iid gaussian noise;
  $\hat{x}^\theta \leftarrow M$ generation from Euler scheme and $g_\theta(z)$;
  $\theta \leftarrow \theta - \beta\text{Adam}\left(\nabla_\theta(\ell^{SK}_{\lambda,c}(d_\varphi(x), d_\varphi(\hat{x}^\theta)\right)$;
end

Algorithm 1: Algorithm for Sinkhorn GAN.

3 Numerical Experiments

We compare the three Euler GAN variants (eGAN , eWGAN and eSGAN) on generated known stochastic processes (namely the Black&Scholes model). Test are done in dimension one in Section 3.1 and scaling study on dimensions up to 20 is done in Section 3.2. In Section 3.3, we propose to use transfer learning technique to enrich a limited historical data set (less than 1000 observations). Our eGan method is trained on Monte Carlo simulations during the first stochastic gradient descent iterations and on real data during the last iteraions. By fine-tuning our model on the real world data set, we show that predictive and risk factor scores based on generated trajectories overcome classical the initial Monte Carlo simulator.

All neural network architectures and hyper parameters such as batch size, learning rate are described in Annex A.3.

3.1 EXPERIMENTS FOR A ONE-DIMENSIONAL SIMULATED PROCESS

In this section, in order to obtain all the useful statistics, we test our approach on a well-known and easy to simulate Black-Scholes model defined by:

$$dX_t = rX_t dt + \sigma X_t dW_t \tag{11}$$

where $r = 0.8$, $\sigma = 0.3$, $X_0 = 0.2$. Simulations are done on 90 dates and maturity is 0.25 (1 simulation per day during 3 months). Table 3.1 compares statistics obtained with the reference MC simulator with those obtained with our eGAN simulators. One can notice a nice adequation for the moments, the 5% and 95% percentile.

Moments	MC	eGAN	eWGAN	eSGAN
		Black-Scholes		
min	1.54e-01	1.61e-01	1.40e-01	1.33e-01
5%q	1.87e-01	1.89e-01	1.90e-01	1.76e-01
mean	2.22e-01	2.22e-01	2.24e-01	2.20e-01
95%q	2.60e-01	2.61e-01	2.57e-01	2.61e-01
max	3.23e-01	3.27e-01	3.01e-01	2.96e-01
var	5.90e-04	5.58e-04	4.55e-04	7.69e-04

Table 1: Moment comparison of real and generated process, where $m(x) = \frac{1}{T}\sum_{t=1}^{T} f(x_t^{(i)})$ with f is min, 5%q,... over batch of size $M = 1000$

The Figure 2 shows a good match between the histograms of the flattened vectors in time of the real and generated time series. These nice results are upgraded by realistic looking trajectories (same Figure). Finally, still in Figure 2, the moments and the envelopes are also confirming the reliablity of our approach.

Thanks to the chosen Euler embedding, drift and volatility estimations can be easily visualized and analyzed. Table 3.1 shows that the (rX_t) seems well represented, while the estimation of the volatility is slightly less accurate.

The temporal dependence between marginals is characterized with AutoCorrelation Function (ACF) in Figure 3. QQPlots and distribution of the log returns show represented in Figure 4.

Time dependence properties compared with real process simulations is evaluate with ACF plots in Figure 3. Mean squared error between auto correlation of the lagged time series measures and real one is satisfying.

Another way to illustrate if we capture previous dates marginal distributions dependencies is to focus on log return which in the Black-Scholes are gaussian. This property seems to be respected by generated time series as illustrated in Figure A.2.

Figure A.2 in the Annex presents percentile of discriminant scores for each real and generated data. The discriminant seeks to maximize the gap between real and fake data, as the generator learns to minimize it.

In Figure 4, QQ-plots are given to compare the Black-Scholes simulated distribution and generated ones. We note here that Sinkhorn Euler GAN presents some weakness compared to other presented GAN, the distribution tails scatter from x-axis. But, all Box-Cox plot demonstrates a good performance of time dependence extrapolation by passing the normality test.

	Real	MC	eGAN	eWGAN	eSGAN
			Black-Scholes		
$\mathbb{E}\left[b_\theta(t, X_t)/X_t\right]$	0.8	0.803	0.838	0.770	0.777
$\mathbb{E}\left[\sigma_\theta(t, X_t)/X_t\right]$	0.4	0.400	0.290	0.400	0.410

Table 2: Comparison of $\mathbb{E}[b_\theta(t, \hat{X}_t^\theta)/\hat{X}_t^\theta]$, $\mathbb{E}[\sigma_\theta(t, \hat{X}_t^\theta)/\hat{X}_t^\theta]$ obtained with eGAN, eWGAN, and eSGAN with the real theoretical value.

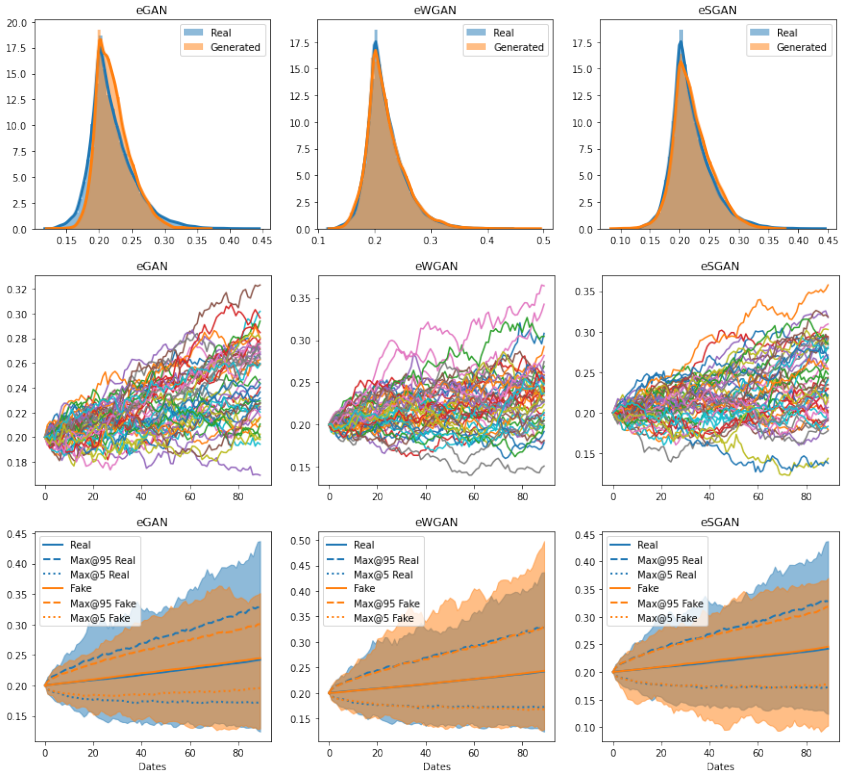

Figure 2: **Up**: Histogram and KDE of real processes (in blue) and GAN generations (in orange) for Black-Scholes. **Middle** :Some fake simulations obtained with the three Euler Gans **Bottom**: Envelope and percentile per Time Step / Comparison with the Monte Carlo reference simulator

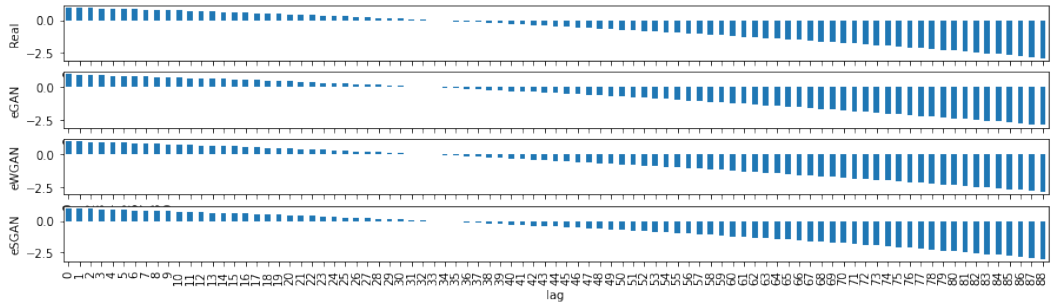

Figure 3: Auto correlation ACF of real and Euler GAN generations

Autocorrelation is particularly accurate, and mean squared error between real BS (first row) and eGAN (resp. eWGAN, eSGAN) is $4.48.10^{-4}$ *(resp.* $1.88.10^{-4}$, $3.59.10^{-4}$).

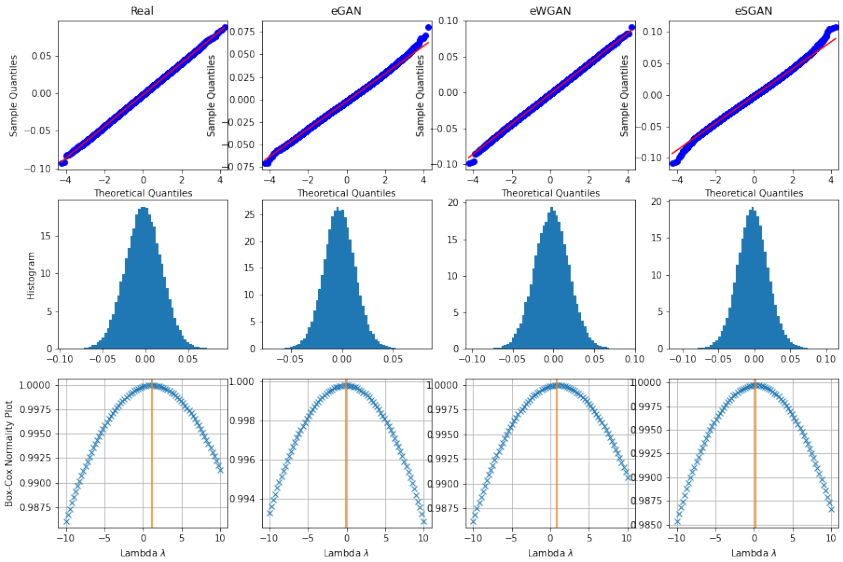

Figure 4: First row, QQplot theorical quantile from real gaussian distribution according to empirical quantiles for log returns of reps. Monte Carlo Black-Scholes simulations, eGAN, eWGAN and eSGAN. The Second row shows distributions of log return, which should be gaussian. Last row indicates BoxCox normality graph.

In Table3 we use our eGANs simulator to price european call and put options. These Monte Carlo pricings are consistent with prices obtained with the theoretical Black&Scholes formula.

Strike	**0.1**	**0.2**	**0.3**	**0.4**
Call Black-Scholes	**0.118**	**0.039**	**0.003**	**0.000**
Call eWGAN	0.119	0.040	0.002	0.000
Call eSGan	0.118	0.039	0.001	0.000
Call eGan	0.117	0.036	0.000	0.000
Put Black Scholes	**0.000**	**0.003**	**0.049**	**0.128**
Put WGAN	0.000	0.003	0.046	0.126
Put eSGan	0.000	0.003	0.047	0.128
Put eGan	0.000	0.001	0.047	0.128

Table 3: Pricing comparison between Call and Put options obtained with the Black-Scholes formula and with discounted average of 1000 simulations obtained from our eGANs ($x_0 = 0.2, r = 0.8, \sigma = 0.4, T = 0.25$)

.

Euler-based generators seems to be good candidates for time series generation. The question we address in the next Section is whether eGAN can scale to higher dimensions.

3.2 Scaling the dimension and capturing the correlation structure

In this section, we use the very same methodology described in section 3.1 but with multivariate time series. We focus on eWGAN and we show that 2-dimensional correlation factor. With a $\rho = 0.6$ two correlated processes are simulated and given for training to the eulerGAN generator. The generator is able to identify the covariance matrix Γ_θ as illustrated in figure 3.2. If the processes are independent, the χ^2 test accepts independence with $\alpha = 0.05$ in dimension 2 and 3. In Figure A.2, a training a 20-dimensional Black-Scholes process with independent increments is done and show an envelope for the 20 process well respected.

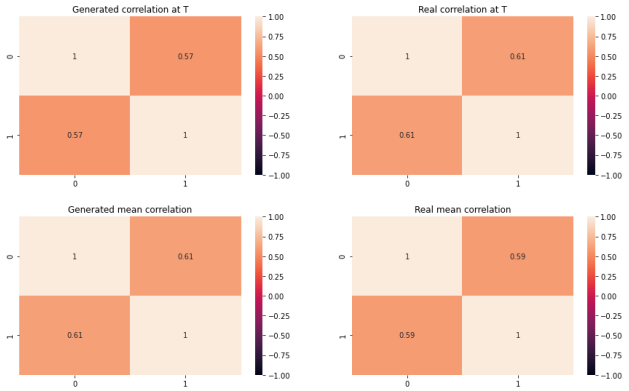

Figure 5: Correlation matrix of 2D Black Scholes. Left column is correlation matrix of generated data from eWGAN and right from Monte Carlo simulations. The first row indicates correlation factors at ending date T, the second is the mean correlation values across time.

3.3 Transfer learning - Monte Carlo to historical tuning

Our Euler GANs may need more data than available to be trained. If one has already designed a reasonable model, we can rely on the use of transfer learning techniques. This transfer learning technique is applied in this section on the S&P500: the networks are firstly trained on a Black-Scholes process calibrated on the S&P500 and secondly on samples of monthly data from 1957 to 1999. We compare percentiles on a testing period (2000-2016) obtained with a Monte Carlo simulator calibrated on the 1957-1999 period and Euler GANs trajectories trained on the MC simulator and on the 1957-1999 dataset. As shown in Table 4 the percentiles obtained with our eGaN is closer to the historical 2000-2016 percentile than the ones obtained with the initial Monte-Carlo simulator. The Value-At-Risk (VaR) (Percentile(X_T - X_{t_0} where $t_0 = 01/12/2016$ and $t_1 = 31/12/2016$)) is also compared in 4 and acknowledge the accuracy of our approach.

Moments	Real data	Monte Carlo	eGAN	eWGAN	eSGAN
VaR	-5.93e-02	-8.26e-02	-1.80e-02	-7.16e-02	-5.58e-02
min	-1.39e-01	-1.82e-01	-3.12e-01	-1.68e-01	-1.58e-01
5%q	-7.65e-02	-8.88e-02	-7.73e-02	-8.41e-02	-7.37e-02
mean	4.70e-03	6.08e-03	3.33e-03	5.95e-03	6.44e-03
95%q	7.13e-02	1.01e-01	8.85e-02	9.61e-02	8.68e-02
max	9.57e-02	1.92e-01	3.09e-01	1.83e-01	1.64e-01
variance	2.01e-03	3.32e-03	2.94e-03	3.01e-03	2.41e-03

Table 4: Average time-flatten moments and Value-at-Risk of real S&P500 data (on testing set), Monte Carlo simulation and Euler GAN generations.

REFERENCES

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

Hans Buehler, Lukas Gonon, Josef Teichmann, and Ben Wood. Deep hedging. *Quantitative Finance*, 19(8):1271–1291, 2019.

Marco Cuturi, Olivier Teboul, and Jean-Philippe Vert. Differentiable ranking and sorting using optimal transport. In *Advances in Neural Information Processing Systems*, pp. 6861–6871, 2019.

Bradley Efron. *The jackknife, the bootstrap and other resampling plans*. SIAM, 1982.

Simon Fécamp, Joseph Mikael, and Xavier Warin. Risk management with machine-learning-based algorithms. *arXiv preprint arXiv:1902.05287*, 2019.

Adeline Fermanian. Embedding and learning with signatures. *arXiv preprint arXiv:1911.13211*, 2019.

Rao Fu, Jie Chen, Shutian Zeng, Yiping Zhuang, and Agus Sudjianto. Time series simulation by conditional generative adversarial net. *arXiv preprint arXiv:1904.11419*, 2019.

Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning generative models with sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, pp. 1608–1617, 2018.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Philippe Jorion. *Value at risk*. McGraw-Hill Professional Publishing, 2000.

Francis A Longstaff and Eduardo S Schwartz. Valuing american options by simulation: a simple least-squares approach. *The review of financial studies*, 14(1):113–147, 2001.

Yonghong Luo, Xiangrui Cai, Ying Zhang, Jun Xu, et al. Multivariate time series imputation with generative adversarial networks. In *Advances in Neural Information Processing Systems*, pp. 1596–1607, 2018.

Jens Schreiber, Maik Jessulat, and Bernhard Sick. Generative adversarial networks for operational scenario planning of renewable energy farms: A study on wind and photovoltaic. In *International Conference on Artificial Neural Networks*, pp. 550–564. Springer, 2019.

Marco Sorge. Stress-testing financial systems: an overview of current methodologies. 2004.

Stefan Steinerberger. Wasserstein distance, fourier series and applications. *arXiv preprint arXiv:1803.08011*, 2018.

Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Tianlin Xu, Li K Wenliang, Michael Munn, and Beatrice Acciaio. Cot-gan: Generating sequential data via causal optimal transport. *arXiv preprint arXiv:2006.08571*, 2020.

Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alche-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 5508–5518. Curran Associates, Inc., 2019. URL `http://papers.nips.cc/paper/8789-time-series-generative-adversarial-networks.pdf`.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

A APPENDIX

A.1 ALGORITHM DETAILS

Input: θ_0, φ_0 randomly chosen, α, β learning rates, K number of iterations, M batch size, n_c critic iterations, c clipping value, $\left(X^{(i)}\right)_{i=1..M}$ real data
Output: θ, φ

$\theta \leftarrow \theta_0, \varphi \leftarrow \varphi_0$;
for $k = 1..K$ **do**
  for $j = 1..n_{critic}$ **do**
    $x \leftarrow M$ samples with $X^{(i)} = (X^{(i)}_{t_1}, \dots, X^{(i)}_{t_N})_{i=1..M}$;
    $z \leftarrow M$ samples iid gaussien noise;
    $\hat{x}^\theta \leftarrow M$ generation from Euler scheme and $g_\theta(z)$;
    $\varphi \leftarrow \varphi + \alpha\text{Adam}\left(\nabla_\varphi(\mathbb{E}[d_\varphi(x)] - \mathbb{E}[d_\varphi(\hat{x}^\theta)]), \alpha\right)$;
    $\varphi \leftarrow \text{clip}(\varphi, -c, c)$;
  end
  $x \leftarrow M$ samples with $X^{(i)} = (X^{(i)}_{t_1}, \dots, X^{(i)}_{t_N})_{i=1..M}$;
  $z \leftarrow M$ samples iid gaussian noise;
  $\hat{x}^\theta \leftarrow M$ generation from Euler scheme and $g_\theta(z)$;
  $\theta \leftarrow \theta - \beta\text{Adam}\left(\nabla_\theta(\mathbb{E}[d_\varphi(\hat{x}^\theta)]), \beta\right)$;
end

Algorithm 2: Algorithm of Euler GAN

Input: θ_0, φ_0 randomly chosen, α, β learning rates, K number of iterations, M batch size, n_c critic iterations, c clipping value, $\left(X^{(i)}\right)_{i=1..M}$ real data
Output: θ, φ

$\theta \leftarrow \theta_0, \varphi \leftarrow \varphi_0$;
for $k = 1..K$ **do**
  for $j = 1..n_{critic}$ **do**
    $x \leftarrow M$ samples with $X^{(i)} = (X^{(i)}_{t_1}, \dots, X^{(i)}_{t_N})_{i=1..M}$;
    $z \leftarrow M$ samples iid gaussien noise;
    $\hat{x}^\theta \leftarrow M$ generation from Euler scheme and $g_\theta(z)$;
    $\varphi \leftarrow \varphi + \alpha\text{Adam}\left(\nabla_\varphi(\mathbb{E}[log(d_\varphi(x))] - \mathbb{E}[d_\varphi(log(1 - \hat{x}^\theta))]), \alpha\right)$;
    $\varphi \leftarrow \text{clip}(\varphi, -c, c)$;
  end
  $x \leftarrow M$ samples with $X^{(i)} = (X^{(i)}_{t_1}, \dots, X^{(i)}_{t_N})_{i=1..M}$;
  $z \leftarrow M$ samples iid gaussian noise;
  $\hat{x}^\theta \leftarrow M$ generation from Euler scheme and $g_\theta(z)$;
  $\theta \leftarrow \theta - \beta\text{Adam}\left(\nabla_\theta(\mathbb{E}[d_\varphi(log(\hat{x}^\theta))]), \beta\right)$;
end

Algorithm 3: Algorithm for adversarial learning for Euler 1-Wasserstein GAN.

A.2 ADDITIONAL FIGURES

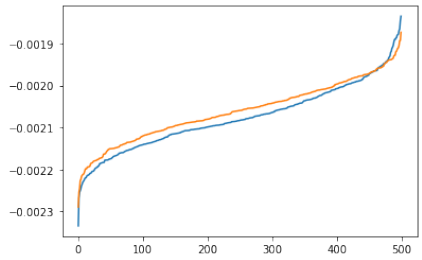

Figure 6: Quantiles of discriminant score distribution. Blue quantile is the real data discriminant score and orange fake ones.

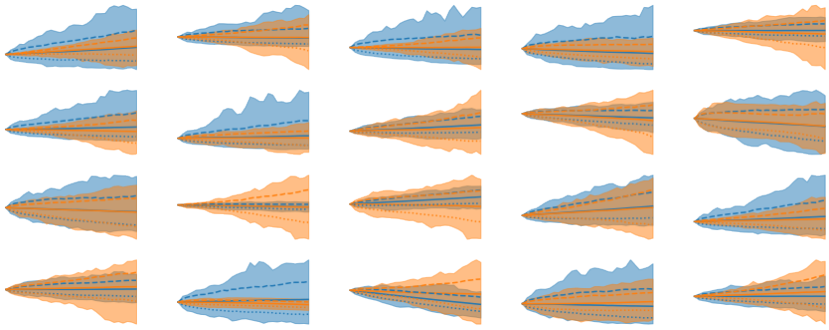

Figure 7: Envelope of a 20-dimensional Black-Scholes (Blue: generated/Orange: Real)

A.3 MODELS AND HYPER-PARAMETERS

Settings of neural networks	
Training data lenght (ndates)	90
Random noise vector dimension	(90×ndim)
Random noise distribution	white noise
Generator structure	{Dense(128),Dense(64),Dense(32),Dense(2×ndim)}
Discriminator structure	{Dense(128),Dense(64),Dense(1×ndim)}
Optimizer	Adam
Learning rates	1.10^{-4}
Batch size	300

Table 5: Neural networks parameters