Japan Journal of Industrial and Applied Mathematics (2023) 40:1359–1390
https://doi.org/10.1007/s13160-022-00534-x

ORIGINAL PAPER

Monte Carlo simulation of SDEs using GANs

Jorino van Rhijn[1] · **Cornelis W. Oosterlee**[2] · **Lech A. Grzelak**[3,4] · **Shuaiqiang Liu**[1]

Received: 22 May 2022 / Revised: 22 May 2022 / Accepted: 28 July 2022 /
Published online: 23 September 2022

Abstract

Generative adversarial networks (GANs) have shown promising results when applied on partial differential equations and financial time series generation. We investigate if GANs can also be used to approximate one-dimensional Itô stochastic differential equations (SDEs). We propose a scheme that approximates the path-wise conditional distribution of SDEs for large time steps. Standard GANs are only able to approximate processes in distribution, yielding a weak approximation to the SDE. A conditional GAN architecture is proposed that enables strong approximation. We inform the discriminator of this GAN with the map between the prior input to the generator and the corresponding output samples, i.e. we introduce a 'supervised GAN'. We compare the input-output map obtained with the standard GAN and supervised GAN and show experimentally that the standard GAN may fail to provide a path-wise approximation. The GAN is trained on a dataset obtained with exact simulation. The architecture was tested on geometric Brownian motion (GBM) and the Cox–Ingersoll–Ross (CIR) process. The supervised GAN outperformed the Euler and Milstein schemes in strong error on a discretisation with large time steps. It also outperformed the standard conditional GAN when approximating the conditional distribution. We also demonstrate how standard GANs may give rise to non-parsimonious input-output maps that are sensitive to perturbations, which motivates the need for constraints and regularisation on GAN generators.

Keywords Generative adversarial networks · Stochastic differential equations · Neural networks · Monte Carlo sampling · Exact simulation · Path-wise conditional distribution

✉ Cornelis W. Oosterlee
c.w.oosterlee@uu.nl

Extended author information available on the last page of the article

Mathematics Subject Classification 60G50 · 60J60 · 60H35 · 65C05 · 68T20

1 Introduction

A significant amount of research has been conducted on generative adversarial networks (GANs), with particularly successful application on image generation problems [1–4]. However, GANs are also found to be notoriously unstable during training [5, 6], while their output is difficult to analyse, although various heuristics have been proposed [7, 8]. Interpreting key properties of GANs explicitly, such as the map learned by the generator, or its output distribution, is typically not possible for image problems. In this work, we propose a sampling scheme for Itô stochastic differential equations (SDEs), where we approximate the path-wise conditional distribution of SDEs with a conditional GAN. The SDE framework allows us to interpret qualities such as the map learned by the generator and the output distribution explicitly, since the flow map between two time steps is available explicitly for some SDEs. We investigate whether our GAN-based scheme can provide a *path-wise* approximation [9] to one-dimensional Itô SDEs. Compared to traditional methods for solving SDEs, the introduction of deep learning-based schemes offers large potential benefits when scaling to higher dimensional problems and overcoming the curse of dimensionality [10, 11]. Our main contributions are as follows:

- We propose a deep learning-based scheme to construct SDE paths for large time steps. A path for any 1D Itô SDE can be sampled by approximating the path-wise conditional distribution with a GAN.
- We propose a 'supervised GAN' to study the input-output map learned by the generator and relate this map to the ability to approximate the SDE path-wise. We show that vanilla GANs may produce non-parsimonious input-output maps that are sensitive to perturbations, motivating the use of constraints on the generator map during training.

1.1 Earlier work

SDEs are prevalent in models of stochastic dynamical systems in engineering, physics, healthcare, and myriad other domains [12]. In finance, they are cornerstone to the modelling of asset prices and interest rates, with applications in portfolio management or the pricing of financial derivatives and related products [13]. In general, the analytical solution to SDEs is not available, which is why practitioners make extensive use of numerical approximations to simulate paths in a Monte Carlo setting [13]. However, a high-quality numerical approximation may be too costly in an *online* setting for practical purposes. At the same time, a continuous representation of the path is not of interest in many applications, but rather the solution at specific times along the path. Through *exact simulation* of an SDE, the exact values of the process underlying the SDE are

sampled at a pre-determined set of times, cf. [14]. However, for general SDEs, exact simulation may not be available. One alternative technique is the *stochastic collocation Monte Carlo* (SCMC) method [15], in which the conditional inverse distribution of an SDE is approximated with a polynomial expansion, e.g. in a Gaussian random variable. Our goal is not to compete with the SCMC algorithm in financial applications, but rather to initiate a new direction for Monte Carlo estimation of SDEs, where the wide applicability of GANs can be demonstrated. The SCMC method only provides an approximation of the conditional distribution given a fixed choice of the time step, the previous value of the process, and the SDE parameters. In [16], this is addressed by combining the SCMC method with a neural network (NN), the 'Seven League Scheme', that predicts the collocation points for the SCMC method, conditional on all model parameters. In our work, the scope is similar, but the conditional distribution will be approximated directly by a conditional GAN, instead of using the SCMC method. This retains the advantage of the Seven League scheme being able to incorporate the dependence on the model parameters and the time step. In addition, if the method can be scaled up to higher dimensions, it exploits the ability of deep learning to combat the curse of dimensionality, where the SCMC method requires the definition of a grid of collocation points [15, 16], which could be expensive in high dimensions.

GANs have been successfully applied on solving (stochastic) PDEs [17–19], however these works rely on application of the PDE operator on outcomes generated with a NN. In the case of Itô SDEs, however, the Brownian motion term precludes differentiability of the dynamics. If the diffusion parameter is constant, Abbati et al. show [20] it is possible to define a measure change that allows one to compute the time derivatives of the transformed random process. This would allow one to 'match' the moments of the time derivative and solve the SDE, but the requirement for constant diffusion processes is too restrictive for the purposes of this work.

Another approach is to apply the 'neural ODEs' by Chen et al. [21] on SDEs. Kidger et al. [22] 'fit' SDEs to time series data, where the SDE coefficients are given by NNs. A GAN architecture is used here as well, where the solution to the SDE defines the output of the 'neural SDE'. The discriminator takes the generated process as input and is itself defined as an SDE, allowing the model to be defined in continuous-time. The model allows the generation of time series data that is equal in distribution to the target, although not necessarily path-wise. We focus on the practical simulation of SDEs, where large time steps are essential and the continuous representation of the process is not of interest.

Instead of focusing directly on solving the SDE, a NN could be used to construct samples that share the same conditional distribution as the target data, which is modelled as a time series, as shown for example in [23–25]. These authors show that the output of their NNs is adapted to the input sequence $\{Z_k\}$ of i.i.d. $N(0, 1)$ random variables, which means that it could find a weak solution to the SDE. However, their approaches would provide no guarantees of finding a strong solution to the SDE, i.e. path-wise approximation given the same Brownian motion on which the SDE is defined. Details about the difference between weak and strong solutions will be further explored in Sect. 2.

In a similar fashion to neural SDEs, in [26] a GAN architecture is proposed that calibrates stochastic local volatility models to market data. This is an example of a data-driven inverse problem using GANs. We, however, will not assume any knowledge about the structure of the SDE.

In this work, we introduce a modified GAN, that we will refer to as a 'supervised GAN', which approximates a strong solution to the SDE. We compare this GAN to the 'standard' conditional GAN, which only yields a weak approximation. Our setting allows us to interpret the conditional output distribution of the generator using non-parametric statistics, as well as the map learned by the generator explicitly. We show that although 'standard' GANs and our modified GAN both approximate the same distribution, their generators may represent very different maps. Meanwhile, the supervised GAN converges faster than the standard GAN during training. Our work motivates the explicit analysis of the map obtained by a model through unsupervised learning, which is relevant in any generative modelling application, from the generation of time series to image generation.

The paper is structured as follows. First, the necessary background behind SDEs and GANs is introduced. Then, the supervised GAN is introduced to allow strong approximation of SDEs, using a training set obtained from the conditional distribution of the SDE. Section 4 shows the key results obtained using our method, followed by a discussion in Sect. 5. Section 6 provides a conclusion and outlook.

2 Preliminaries about SDEs and GANs

In this section, we discuss the preliminaries underlying SDEs and GANs, notably weak and strong solutions, discrete-time approximation and conditional GANs.

2.1 SDE definition

Suppose we are given a probability space $(\Omega, \mathcal{F}, P)$ and let $\{W_t\}_{t\geq 0}$ be a standard Brownian motion on $\mathbb{R}$, adapted to its natural filtration $\mathcal{F}_t := \sigma\big(\{W_s : s \leq t\}\big)$. A one-dimensional SDE of the Itô type is then defined as follows [12]:

$$dS_t = A\big(t, S_t\big)dt + B\big(t, S_t\big)dW_t, \quad S_0 \in \mathbb{R}, \tag{1}$$

where $\{S_t\}_{t\geq 0}$ is a continuous-time random process on $\mathbb{R}$ adapted to $\mathcal{F}_t$. $A(t, S_t)$ and $B(t, S_t)$ are themselves $\mathcal{F}_t$-measurable random processes on $\mathbb{R}$. One could write the SDE equivalently in its Itô integral form, as follows [12]:

$$S_t = S_0 + \int_0^t A\big(\tau, S_\tau\big)d\tau + \int_0^t B\big(\tau, S_\tau\big)dW_\tau, \quad \forall t \geq 0 \quad P\text{-a.s.} \tag{2}$$

From now on, we write a random process $\{\cdot\}_{t\geq 0}$ succinctly as $\{\cdot\}$. We will refer to a realisation of the process $\{S_t\}$ over a finite time period as a *path*. Note that a path is completely defined once the Brownian motion $\{W_t\}$ has taken a realisation on the respective time interval. The nature of $\{S_t\}$ as a random process complicates the

notion of the existence and uniqueness of the solution of an SDE. Suppose that $\{S_t\}$ is a solution to Eq. (1). A solution is called *path-wise unique* if the following holds for any other $\mathcal{F}_t$-adapted solution $\{S'_t\}$ [27]:

$$P\left(S_t = S'_t\right) = 1, \tag{3}$$

i.e. the paths corresponding to the solution are equal P-a.s. We distinguish between a *strong solution* and a *weak solution*. If we are given a Brownian motion $\{W_t\}$, a strong solution is the path-wise unique solution of Eq. (1) corresponding to that Brownian motion. A weak solution also satisfies Eq. (1), but may be defined with respect to a different Brownian motion than $\{W_t\}$ or even a different probability space. A solution is called *weakly unique* if it is equal in law to any other solution $\{S'_t\}$: $S_t \overset{\mathcal{L}}{=} S'_t$ [27]. Both weak and strong solutions are weakly unique, but only a strong solution is path-wise unique [12]. A unique strong solution exists if $A(t, S_t)$ and $B(t, S_t)$ satisfy Lipschitz conditions, if S_0 is independent of $\mathcal{F}_t$ and if the process $\{S_t\}$ is square-integrable for all t, see [12] for details. A sufficient condition for a weak solution is that $A(t, S_t)$ and $B(t, S_t)$ must be bounded and continuous and $|B(t, S_t)| \geq \varepsilon > 0$ for some positive real ε [13]. The often cited conditions for weak and strong solutions are sufficient, but not necessary, as is clear from multiple examples of SDEs that do not satisfy the conditions, but still have a strong solution [12, 13]. In the following, we will restrict ourselves to SDEs for which a strong solution exists.

2.2 Discrete-time schemes

It is possible to approximate Eq. (2) by a discrete-time scheme, based on a stochastic Taylor expansion, such as the Euler or Milstein schemes [13]. Recall that in the 1D case, the Euler and Milstein schemes are given by [13]:

$$\tilde{S}_{t+\Delta t} = \tilde{S}_t + A\left(t, \tilde{S}_t\right)\Delta t + B\left(t, \tilde{S}_t\right)\Delta W_t + \zeta\left[\frac{1}{2}B\left(t, \tilde{S}_t\right)B'\left(t, \tilde{S}_t\right)\left(\Delta W_t^2 - \Delta t\right)\right], \tag{4}$$

where $\zeta = 0$ for the Euler and $\zeta = 1$ Milstein scheme. Δt is the time step of the discretisation, $\Delta W_t := W_{t+\Delta t} - W_t$ and $B' := \frac{\partial B}{\partial S_t}$. We denote the discrete-time approximation of S_t by $\tilde{S}_t$. A key property of these schemes is that they approximate the strong solution $\{S_t\}$ of an SDE, if it exists [13]. In order to quantify their accuracy, we define the *weak error* e_w and the *strong error* e_s as follows, for $t \geq 0$:

$$e_w := \left|\mathbb{E}f\left(S_t\right) - \mathbb{E}f\left(\tilde{S}_t\right)\right|, \tag{5}$$

$$e_s := \mathbb{E}\left|S_t - \tilde{S}_t\right|, \tag{6}$$

where f is some real-valued polynomial function. Note how the weak error describes how much the approximation differs in distribution, i.e. how it differs from a weakly unique solution, while the strong error indicates how much the approximation differs path-wise from the strong solution. The convergence rate of a discrete-time

scheme can be expressed in terms of Δt: the weak error of both the Euler and Milstein schemes can be shown to be of $O(\Delta t)$, while the strong error is of $O(\sqrt{\Delta t})$ for the Euler scheme and of $O(\Delta t)$ for the Milstein scheme [13].

2.3 Generative adversarial networks

A GAN is a combination of two NNs that are trained adversarially, cf. [1]. During training, the *generator* network iteratively maps a prior input to a new random sample, while the *discriminator* network alternatingly receives either a sample from the generator or the training set of reference samples. The discriminator assigns a score on [0, 1] to the input it receives. The input samples to the discriminator are labeled either 0 ('fake', coming from the generator) or 1 ('real', coming from the training set).[1] The output of the discriminator can be interpreted as the confidence it assigns to the input being 'real'. Suppose that the generator G_θ is parameterised by $\theta \in \mathbb{R}^p$ and the discriminator D_α is parameterised by $\alpha \in \mathbb{R}^q$, for some $p, q \in \mathbb{N}$. The GAN objective function can then be defined in terms of both G_θ and D_α as follows:

$$V(G_\theta, D_\alpha) = \mathbb{E}_{X\sim P^*}\big[\log D_\alpha(X)\big] + \mathbb{E}_{Z\sim P_Z}\big[\log\big(1 - D_\alpha \circ G_\theta(Z)\big)\big], \tag{7}$$

where P^* is the target distribution associated with the training data and P_Z is the prior distribution from which input samples to the generator are drawn. '$\circ$' denotes the composition of functions. The value function captures the degree to which the discriminator succeeds in recognising real samples (first term) and recognising 'fake' samples (second term). From the generator's point of view, this is the other way around and the second term is inversely related to its performance. The roles of the generator and discriminator give rise to the following adversarial objective:

$$\inf_\theta \sup_\alpha V(G_\theta, D_\alpha), \tag{8}$$

Note the resemblance with a two-player zero-sum game and minimax theory [1, 28]. It can be shown that a solution to Eq. (8) coincides with equality in distribution between the target P^* and generator output distribution P_θ [1, 29].

The generator and discriminator are each given their own loss function, based on Eqs. (7) and (8):

$$L_D = -\mathbb{E}_{X\sim P^*}\big[\log\big(D_\alpha(X)\big)\big] - \mathbb{E}_{Z\sim P_Z}\log\big(1 - D_\alpha \circ G_\theta(Z)\big), \tag{9}$$

$$L_G = \mathbb{E}_{Z\sim P_Z}\log\big(1 - D_\alpha \circ G_\theta(Z)\big), \tag{10}$$

for which the minima are found with a suitable gradient descent algorithm. Equation (10) tends to give vanishing gradients during training, which is why it is often replaced by $L_G = -\mathbb{E}_{Z\sim P_Z}\log\big(D_\alpha \circ G_\theta(Z)\big)$ [5]. We adopt this modification as well.

[1] Note that it is possible to define alternative labels, e.g. 0.1 and 0.9, or smooth variants, which sometimes improves results in practice [7].

We will refer to the GAN described so far as the 'vanilla GAN', as it forms the basis for further extensions. One key extension is the *conditional GAN*, introduced in [30]. In this architecture, the generator and discriminator receive a vector with conditional information as an additional input, which allows the GAN to learn how the output should vary based on a condition label, e.g. generating images of apples or oranges based on the given input. Aside from the appearance of the conditional label, the loss functions remain unchanged. If we let $y \in \mathbb{R}^{n_c}$ be a vector with n_c conditional inputs, the joint objective function becomes [30]:

$$\inf_{\theta} \sup_{\alpha} \left[\mathbb{E}_{X|y\sim P^*} \log D_\alpha(X \mid y) + \mathbb{E}_{Z\sim P_Z} \log \left(1 - D_\alpha \circ G_\theta(Z \mid y)\right)\right], \tag{11}$$

with similar expressions for the loss functions as in Eqs. (9) and (10).

2.4 The generator as a parametric map

The GAN is part of a class of methods with for approximating the distribution of a target random variable $X \sim P_X$, starting from a prior $Z \sim P_Z$. Let us assume that both $X, Z \in \mathbb{R}^n$. The model that approximates the target is defined by a map $\varphi_\theta : \mathbb{R}^n \to \mathbb{R}^n$, $z \mapsto \varphi_\theta(z)$, with parameter set $\theta \in \mathbb{R}^p$, for some integer dimensions n, p. Let us assume the distribution of $\varphi_\theta(Z)$ is given by P_θ, i.e. the distribution of the output samples is P_θ. The goal is then to change the parameters θ such that $P_\theta \overset{d}{\approx} P_X$. In our case, the role of φ_θ is taken by the GAN generator. We can use common methods to quantify the 'difference' between the distributions P_θ and P_X, such as the Jensen-Shannon (JS) divergence [31]. This quantity is defined for any two absolutely continuous distribution measures P and Q through the Kullback-Leibler (KL) [32] divergence as:

$$JS(P\|Q) = \frac{1}{2}(KL(P\|M) + KL(Q\|M)), \tag{12}$$

$$KL(P\|Q) = \int_{\mathcal{X}} \log\left(\frac{p(x)}{q(x)}\right) q(x)dx, \tag{13}$$

where p and q are the densities associated with respectively P and Q, $M = \frac{P+Q}{2}$ and $\mathcal{X} \subseteq \mathbb{R}^n$ is the support of both distributions. It can be shown that $JS(P\|Q) \geq 0$, with equality iff $P = Q$ [31]. The goal is to choose θ such that it minimises $JS(P_\theta\|P_X)$. This could be done with standard techniques such as stochastic gradient descent (SGD) [33]. However, we now focus on how the map φ_θ relates to the induced distribution P_θ. In typical problems, this map is not of interest, such as in image generation problems, where no reasonable model exists for φ_θ, making it very difficult to draw conclusions based on the learned map φ_θ. However, in this work, we study the map φ_θ explicitly, which is required for our strong approximation criterion. Meanwhile, it enables us to make qualitative statements about the map learned by the GAN generator.

Let us turn to a simple example where we can write the map from Z to X explicitly: the lognormal distribution.

Example 2.1 (Lognormal distribution) Let $X, Z \in \mathbb{R}$ and let $X = e^Z$ with $Z \sim N(0,1)$. One function that minimises $JS(P_\theta \| P_X)$ is given by $\varphi^+ := e^Z$. However, it is not unique, as we could have equally chosen $\varphi^- := e^{-Z}$ by symmetry of the normal distribution. Both choices yield a JS divergence of exactly 0 and we may expect an SGD-based algorithm to find either of the solutions in some proportion.

If the lognormal distribution in Example 2.1 was approximated by a NN with infinite capacity, i.e. one which can represent any continuous map $\varphi : \mathbb{R}^n \to \mathbb{R}^n$, including φ^+ or φ^-, the set of maps minimising the JS divergence would be $\{\varphi^+, \varphi^-\}$. Note that in n dimensions, i.e. $\boldsymbol{X} = [e^{Z_1}, e^{Z_2}, \dots, e^{Z_n}]^T$, the set of candidate functions with JS divergence exactly 0 grows as 2^n, as each $Z_i \sim N(0,1)$ is individually symmetric about the origin.

In reality, NNs do not have infinite capacity, but the set of maps they can represent is restricted by the parameter set $\Theta \subseteq \mathbb{R}^p$. This means that in general, $JS(P_\theta \| P_X) > 0$. The parametric map φ_θ may only be able to bring the JS divergence down to some $\varepsilon > 0$. The key question we are interested in is how many maps lie within an ε from optimality, and how different these maps are from the 'true' optimum, e.g. φ^+ or φ^- in Example 2.1. Let us define the collection of maps that lie within an ε from optimality in terms of the JS divergence as:

$$\mathcal{V}_\Theta^\varepsilon := \left\{ \varphi_\theta : \left(0 < JS(P_\theta \| P_X) < \varepsilon\right), \theta \in \Theta \right\}, \tag{14}$$

where we stress the dependence on ε and Θ. Since we did not make any assumptions on φ_θ, the class of functions $\mathcal{V}_\Theta^\varepsilon$ found after applying SGD may be very large. The number of elements of $\mathcal{V}_\Theta^\varepsilon$ should increase with ε, as more maps give rise to distributions that lie within an ε of P_X.

In addition to the finite capacity of the parameter set $\theta \in \Theta$, NNs are trained on finite datasets, not perfectly representing P_X. Thus, even if we know the 'true' underlying map $\varphi^*(Z)$ from which a dataset X was constructed, we may find a map $\varphi_\theta \in \mathcal{V}_\Theta^\varepsilon$ with a gradient descent algorithm that is very different from φ^*. In other words, maps that are close in JS divergence may not be close in function space. Note that although we chose the JS divergence to illustrate the point, we could define similar classes of functions for other divergence measures, such as the KL divergence, total variation distance, etc. The JS divergence is particularly relevant in the case of GANs, as one can show that the optimal generator in Eq. (8)—given the optimal discriminator - minimises the quantity $JS(P_\theta \| P^*)$ [1, 29].

The key observation is that algorithms minimising a distributional quantity or metric do not impose any restrictions on the map φ_θ. It may for example be highly non-smooth in regions along its support, even though the approximation in distribution is close. Therefore, although it is typically not tractable to study φ_θ, we argue that qualitative properties of φ_θ should be of interest in generative modelling, given its implications for the robustness of the resulting sampling scheme.

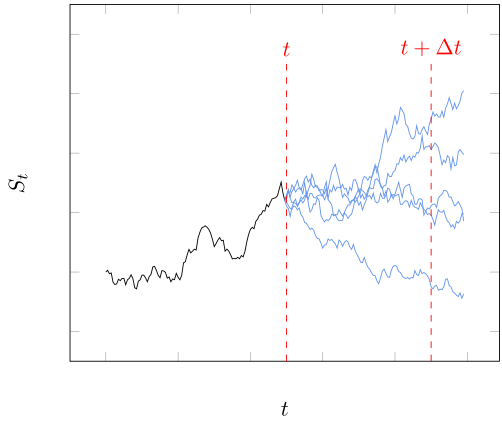


Fig. 1 Illustration of the problem setting: given the process $\{S_t\}$ up to time t, obtain samples from the process at time $t + \Delta t$ using a GAN

3 Methodology

Let $\{S_t\}$ be the strong solution to an SDE as defined in Equation (1). Suppose we are interested in the solution at N times, i.e. $0 = t_0 < t_1 < \cdots < t_N = T$. Let us denote the conditional distribution function of the solution at time t_k given the previous sample by $F_{S_{t_k}|S_{t_{k-1}}}$ for $k \in \{1, \ldots, N\}$. When using exact simulation, one samples iteratively from the distribution of $S_{t_k} \mid S_{t_{k-1}}$, cf. [14]. This is possible due to the *Markov property* of Itô SDEs [27], i.e. each $S_{t_k} \mid S_{t_{k-1}}$ is independent of $\mathcal{F}_{t_{k-1}}$ for $k \in \{1, \ldots, N\}$. This allows one to construct a path $\{S_{t_0}, S_{t_1}, \ldots, S_{t_N}\}$ along the time discretisation by iterated sampling from the distribution of $S_{t_k} \mid S_{t_{k-1}}$. Let us from now on assume, without loss of generality, that our discretisation consists of time steps of equal size Δt. Paths can then be constructed by iteratively sampling from the conditional distribution of $S_{t+\Delta t} \mid S_t$, having initialised the process at some $S_0 \in \mathbb{R}$ at $t = 0$, which is illustrated in Fig. 1.

If the conditional distribution of $S_{t+\Delta t} \mid S_t$ is approximated with a conditional GAN, i.e. conditional on Δt and S_t, new points can be sampled iteratively along the path with the conditional GAN as follows:

$$\hat{S}_{t+\Delta t} \mid \hat{S}_t = G_\theta\big(Z, \Delta t, \hat{S}_t\big), \tag{15}$$

where $Z \sim N(0, 1)$. We denote the approximation of S_t by the GAN by $\hat{S}_t$. This could be further generalised by conditioning on the SDE parameters contained in $A(t, S_t)$ and $B(t, S_t)$ as well. In this work, we will hold them fixed and train the conditional GAN on a dataset of tuples $((S_{t+\Delta t} \mid S_t), S_t, \Delta t)$ with varying Δt and S_t.

3.1 Supervised GAN

Since $S_{t+\Delta t} \mid S_t$ is a continuous random variable and $F_{S_{t+\Delta t}|S_t} \sim U(0, 1)$, for each realisation of $S_{t+\Delta t} \mid S_t$, there is a unique realisation of $U \sim U(0, 1)$. Let F_Z denote the cumulative distribution function of the random variable $Z \sim N(0, 1)$. Both

$F_{S_{t+\Delta t}|S_t}$ and F_Z are strictly increasing, as they are based on continuous random variables. Therefore, their distribution functions are bijections and their inverses exist. Thus, for each realisation of $\omega \in \mathcal{F}_t$ and corresponding realisation of the process $\big(S_{t+\Delta t} \mid S_t\big)(\omega)$, there is a unique realisation $U(\omega)$. In turn, for each $U(\omega)$ there is a unique realisation $Z(\omega)$. These realisations are related as follows:

$$\big(S_{t+\Delta t} \mid S_t\big)(\omega) = F^{-1}_{S_{t+\Delta t}|S_t}\big(F_Z(Z(\omega))\big). \tag{16}$$

In the SCMC scheme, Eq. (16) is approximated by a polynomial expansion in Z [15, 16]. This allows path-wise comparison of the SCMC scheme to e.g. a Milstein scheme in [16], where $Z(\omega)$ is used to define the Brownian motion increment between time t and $t + \Delta t$. In this work, we approximate the conditional inverse function directly using the generator of a GAN. However, as we saw in Sect. 2.4, an approximation to the target distribution does not imply the underlying map is unique. For a strong approximation to the process, we must approximate Eq. (16). That is, we must ensure that we preserve the relation between the prior $Z(\omega)$ and $(S_{t+\Delta t} \mid S_t)(\omega)$. To this end, we can build a training set of samples $(Z, (S_{t+\Delta t} \mid S_t), S_t, \Delta t)$ as input to both the generator *and* discriminator, where Z is found using the 'inverse' of Eq. (16):

$$Z(\omega) = F_Z^{-1}\Big(F_{S_{t+\Delta t}|S_t}\big(S_{t+\Delta t} \mid S_t\big)(\omega)\Big). \tag{17}$$

This way, we do not only let the GAN learn the distribution $F_{S_{t+\Delta t}|S_t}$, but the map from $Z(\omega)$ to $\big(S_{t+\Delta t} \mid S_t\big)(\omega)$, which carries the information about the event $\omega \in \mathcal{F}_t$ that corresponds to the realisation of the specific value $\big(S_{t+\Delta t} \mid S_t\big)(\omega)$. We will call this architecture the 'supervised GAN', as it is a GAN-based equivalent to training a standard feed-forward network on the mean squared error between $F^{-1}_{S_{t+\Delta t}|S_t}\big(F_Z(Z(\omega))\big)$ and $G_\theta(Z(\omega), S_t, \Delta t)$. The supervised GAN discriminator receives as input $\big(Z(\omega), \big(S_{t+\Delta t} \mid S_t\big)(\omega), S_t, \Delta t\big)$, while the vanilla GAN discriminator only receives $\big(\big(S_{t+\Delta t} \mid S_t\big)(\omega), S_t, \Delta t\big)$ but not $Z(\omega)$. This constrains which input-output map from the generator is allowed. It allows the supervised GAN to perform a path-wise approximation, while the vanilla GAN is only guaranteed to approximate the process in conditional distribution (e.g. may fail to distinguish the output given $-Z$ from $+Z$), yielding a weak approximation.

3.2 Analysis of the output distribution

In order to quantify the difference between the conditional distribution of the generator output and $F_{S_{t+\Delta t}|S_t}$, we use two non-parametric statistics: the *Kolmogorov–Smirnov* (KS) statistic and the *Wasserstein distance* in 1D. The 2-sample KS statistic is defined as follows, cf. [34]:

$$u_n := \sup_x \big|F_X(x) - F_Y(x)\big|, \tag{18}$$

where F_X and F_Y are two empirical cumulative distribution functions (ECDFs), one of which corresponding to the generator output and the other to the reference distribution. Note that if the CDF of the target distribution is available analytically, we can use it in Equation (18) instead of its ECDF.

In 1D, the r-Wasserstein-distance (i.e. based on the r-norm) between two distribution functions F_X and F_Y can be expressed as follows, for some $r \in \mathbb{R}^+$ [35]:

$$v_r(F_X, F_Y) = \left(\int_0^1 |F_X^{-1}(x) - F_Y^{-1}(x)|^r dx \right)^{\frac{1}{r}}, \tag{19}$$

where $F_{(\cdot)}^{-1}$ denotes the inverse distribution function, i.e. the quantile function of the random variable under consideration. In this work, we will set $r = 1$ and use the 1-Wasserstein distance. F_X and F_Y are computed from a dataset of samples $\{\hat{X}_i\}_{i=1}^n$ obtained through inference with the GAN and a dataset $\{X_i\}_{i=1}^n$ drawn from the reference distribution.

The KS-statistic computes the largest difference between two (E)CDFs, i.e. 'vertical differences' in the plane $F_X(x)$ versus x. In 1D, the Wasserstein distance can be thought of as the average distance between the quantiles of two distributions, i.e. 'horizontal differences' [35]. The combination of these statistics simultaneously thus allows us to capture different features of both distributions.

The challenge is, however, to interpret the value of both statistics given a sample of size N_{test} of both the reference distribution and the GAN output. We could construct a reference value by drawing two i.i.d. vectors of size N_{test} containing realisations of the reference distribution, say $X, Y \overset{iid}{\sim} F_{S_{t+\Delta t}|S_t}$. If we choose N_{test} too low (e.g. 100), the approximation of the distribution function will be very coarse and both statistics would exhibit a large variance. If we choose N_{test} high, e.g. 10^5, the KS statistic and Wasserstein distance of this reference value will tend towards 0. This dependence on N_{test} makes comparison between the statistics on the GAN output and reference value difficult. In order to avoid a particular choice of N_{test}, we will compute the statistics for a range of values of N_{test}, e.g. $\{100, 1000, \ldots, 10^5\}$ and plot the result versus N_{test}. We will repeat this experiment on a set of N_{test} samples obtained with a single-step Euler and Milstein approximation, based on the same time step Δt and 'starting value' S_t that the GAN is tested on.

3.3 Data pre-processing

In our setting, the only knowledge of the process $S_{t+\Delta t} \mid S_t$ that we assume to be available are the SDE parameters and the latest value of the process S_t. We can leverage the fact that the sample S_t is available, by training the network on the *relative* increase of $S_{t+\Delta t} \mid S_t$ compared to S_t, instead of its absolute value. This way, the NN does not need to learn where to place the distribution for each S_t, but automatically outputs a distribution in a neighbourhood of S_t. Following [23], we use logreturns and let the conditional GAN approximate the logreturns-transformed process:

$$R_{t+\Delta t} \mid S_t := \log\left(\frac{S_{t+\Delta t} \mid S_t}{S_t}\right). \tag{20}$$

The approximation of the process $S_{t+\Delta t} \mid S_t$ is then obtained with the inverse transform:

$$\hat{S}_{t+\Delta t} \mid S_t = S_t e^{G_\theta(Z, S_t, \Delta t)}. \tag{21}$$

Using logreturns comes with the additional benefit of centering the distribution near the origin. NNs typically converge faster if the training set is standardised [36], i.e. if the inputs to the network are of mean zero and unit variance. The variance will still vary with the model parameters, and as we do not assume that the moments of the target distribution are known, we cannot simply standardise the dataset and invert the standardisation step after training. Moreover, financial SDEs are typically heavy-tailed, which makes standardisation with point estimates ineffective.

The logreturns transformation comes with a complication for SDEs that can reach values arbitrarily close to zero, such as the CIR process [9, 37]. This means that the numerator and denominator in Eq. (20) can differ by many orders of magnitude (e.g. a sample starting at 0.1 and jumping to 10^{-6} and vice-versa), which leads to large and potentially unbounded output domains after the logreturns transform, which is undesirable. Therefore, an SDE that can jump to and from values near the origin should be pre-processed in a different way. Since we assume the model parameters are known, one could use this knowledge as an alternative to standardisation. For example, the CIR process reverts to a long-term mean $\bar{S}$, which is assumed to be known. We use this parameter to shift and scale the distribution as follows:

$$R_{t+\Delta t} \mid S_t := \left(S_{t+\Delta t} \mid S_t - \bar{S}\right)/\bar{S}, \tag{22}$$

which is approximated with the conditional GAN. The approximation of $S_{t+\Delta t} \mid S_t$ is then obtained by inverting Eq. (22). Since values of the process can get arbitrarily close to zero, the generator may output negative values very close to 0. We 'rectify' the output by taking the absolute value of the generator output: $|(R_{t+\Delta t} \mid S_t + 1)\bar{S}|$, making sure the final approximation of the process is in $\mathbb{R}^+$.

3.4 Network architecture

The generator and discriminator are both implemented as feed-forward NNs, using 4 hidden layers and 200 neurons per layer. A LeakyReLU activation (i.e. $x \mapsto \max(x, 0) + a \min(x, 0)$, for some $a \in \mathbb{R}$) [38] is chosen as the non-linearity after each layer, except the output layers of the generator and discriminator. This activation is chosen, since the distribution of the inputs and hidden state of the network is heavy-tailed. Saturating activations, such as the tanh function, were therefore found to be less effective. The discriminator is given a logistic function at the output, to force the output to be on [0, 1]. The generator has no output activation. All implementations are made using PyTorch [39] and run on an NVIDIA RTX 2070

Super GPU with 8 GB of memory. See Appendix 1 for more details on the architecture and training process.

4 Results

To assess the GAN, we study three different properties that allow us to compare the vanilla GAN to the supervised GAN. Firstly, we study the ability of both GANs to approximate the conditional distribution $F_{S_{t+\Delta t}|S_t}$, for several test values of Δt and S_t. Secondly, we compute the weak and strong error of artificial paths constructed with the vanilla GAN and supervised GAN. Thirdly, we explicitly study the map from prior sample Z to the sample $S_{t+\Delta t} \mid S_t$ learned by the generator for both GANs.

4.1 SDEs under consideration

To test the supervised GAN, we choose two common SDEs that have a strong solution: *geometric Brownian motion* (GBM) [40] and the *Cox-Ingersoll-Ross* (CIR) process [37]. The dynamics are given by:

$$\text{GBM}: \quad dS_t = \mu S_t dt + \sigma S_t dW_t, \tag{23}$$

$$\text{CIR}: \quad dS_t = \kappa\left(\bar{S} - S_t\right)dt + \gamma\sqrt{S_t}dW_t, \tag{24}$$

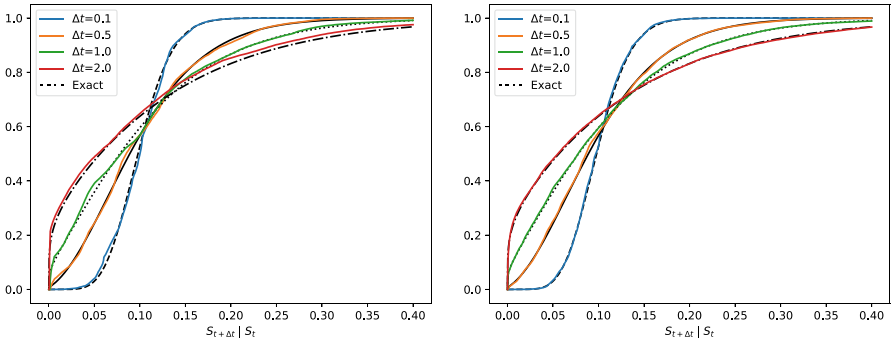
where μ and σ of GBM denote respectively the drift and volatility of the underlying asset. κ controls the rate at which the CIR process reverts to its long-term mean $\bar{S}$, while γ represents the volatility of the CIR process.

The conditional distribution of $S_{t+\Delta t} \mid S_t$ is available explicitly for both SDEs, which allows the construction of an exact training set and simplifies the interpretation of the results. In the GBM case, application of Itô's lemma on the process $\log S_t$ allows one to immediately derive that the solution is lognormally distributed [40, p. 226]. For the CIR process, one can show that $S_{t+\Delta t} \mid S_t$ follows a scaled non-central χ^2-distribution with some non-centrality parameter ξ, degrees of freedom δ and scaling factor $\bar{c}$ [37]:

$$S_{t+\Delta t} \mid S_t \sim \bar{c}\, \chi^2(\xi, \delta), \tag{25}$$

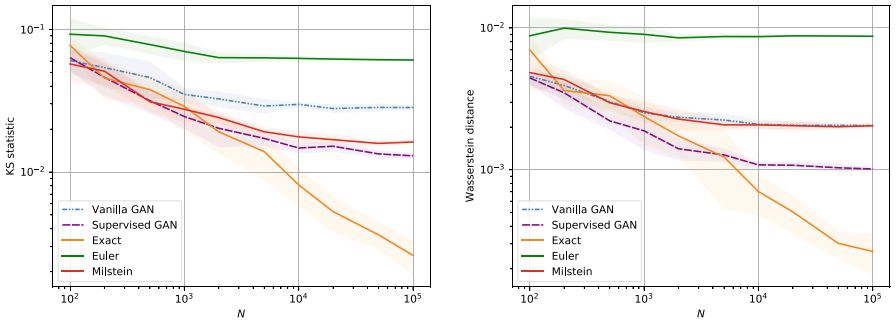
where $\bar{c}, \xi$ and δ are expressed in terms of the SDE parameters [9], see Appendix 3 for details. The presence of the square root in Eq. (24) introduces a complication when approximating the SDE with a discrete-time scheme, which could take negative values. Therefore, we apply a modified, truncated version of the Euler [41] and Milstein [42] schemes on the CIR process, see Appendix 3 for details.

If $\delta < 2$, the non-central χ^2-distribution exhibits near-singular behaviour in a region near zero, i.e. $(0, q]$ for arbitrarily small $q > 0$, allowing the process to 'hit' zero [37]. If $\delta \geq 2$, the process does not exhibit this property and remains strictly positive. This regime for δ is known as the *Feller condition* [43]. For our numerical experiments, we chose two regimes of parameters, one in which the Feller condition



(a) Vanilla GAN **(b)** Supervised GAN

Fig. 2 ECDF plots of the vanilla and supervised GAN output with $S_t = 0.1$ and $\Delta t \in \{0.1, 0.5, 1, 2\}$



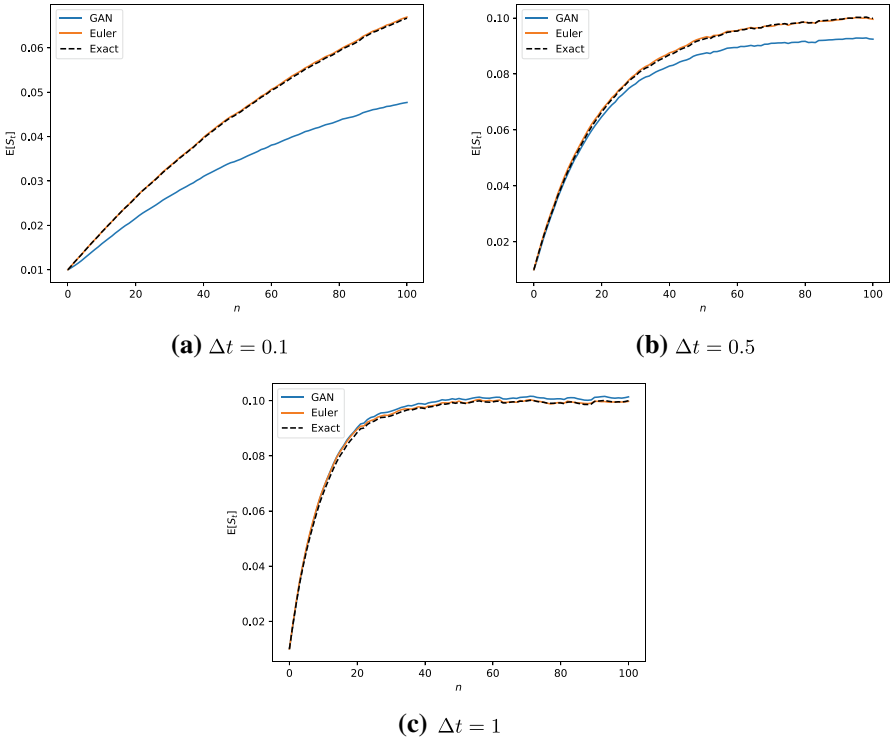
(a) KS statistic **(b)** Wasserstein distance

Fig. 3 KS statistic and Wasserstein distance at $\Delta t = 0.4$, versus the size of the test set. The confidence bands show the standard deviation based on 10 repetitions of the experiment, i.e. 10 i.i.d. samples of N random inputs to both GANs. The mean of both statistics is reported in the solid and dashed lines

is satisfied and one in which it is violated. The near-singular behaviour of the distribution makes the latter case the most challenging.

4.2 Approximating the conditional distribution

We focus on the CIR dynamics for which the Feller condition is violated. The results for GBM and the case where the Feller condition is satisfied are provided in Appendix 2. First, we present the distribution of the output of the vanilla and supervised GAN in Fig. 2, which shows the ECDF of $\hat{S}_{t+\Delta t} \mid S_t$ for fixed $S_t = 0.1$ and four choices of Δt. We compare this with the exact distribution given in black. We see that both GANs adapt the shape of the output distribution to match



(a) $\Delta t = 0.1$ **(b)** $\Delta t = 0.5$

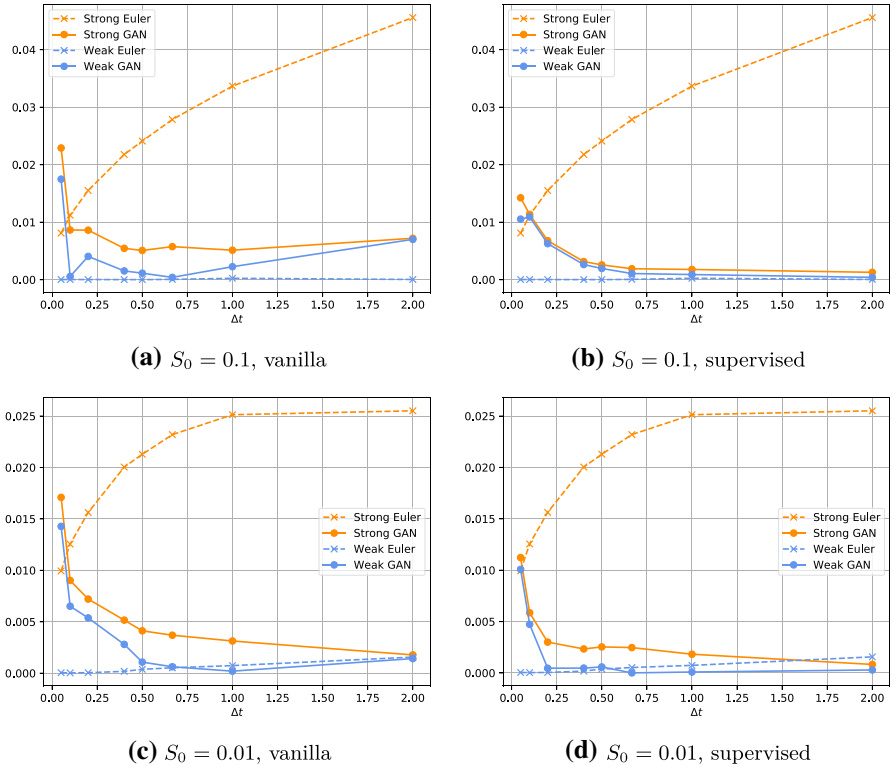
(c) $\Delta t = 1$

Fig. 4 Mean of 10^5 paths obtained with the supervised GAN after n repetitions of $G_\theta(Z, S_t, \Delta t)$, starting from $S_0 = 0.01$. The mean reversion parameter was set to $\bar{S} = 0.1$. The paths generated by the supervised GAN indeed exhibit mean reversion, although the GAN does not revert to the correct mean for every Δt. As Δt decreases, the error in the mean to which the GAN samples revert increases. This shows that the approximation of the conditional distribution is less accurate for smaller Δt, in line with our remaining benchmarks

the exact distribution, while the supervised GAN appears to provide a more accurate approximation.

In Fig. 3, the KS statistic and Wasserstein distance are reported for a range of test sizes, using the method described in Sect. 3.2. S_t was set to 0.1. Δt was chosen to be 0.4 in Fig. 3, for which the KS statistic was close to the Milstein scheme for the supervised GAN. For $\Delta t > 0.4$, both statistics of the supervised GAN were lower than those of the Milstein scheme, i.e. the supervised GAN outperforms both the Euler and Milstein schemes for $\Delta t > 0.4$. The supervised GAN outperforms the vanilla GAN on both statistics. Similar plots can be made for different choices of Δt and S_t and similar results were found for GBM and the case where the Feller condition was satisfied.

Figures 2 and 3 provide a 'snapshot' of the output of both GANs for one or more conditional inputs. For the CIR process, we can test a qualitative property that requires the GAN to accurately capture the conditional dependence on Δt and S_t. We show this for the supervised GAN. The CIR process reverts in the mean to the



(a) $S_0 = 0.1$, vanilla **(b)** $S_0 = 0.1$, supervised

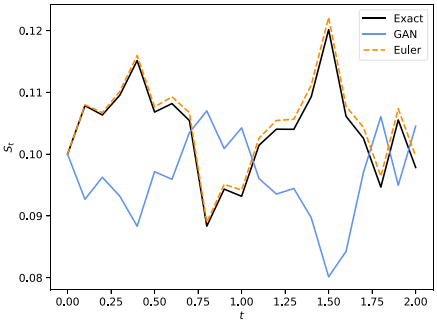
(c) $S_0 = 0.01$, vanilla **(d)** $S_0 = 0.01$, supervised

Fig. 5 Weak and strong errors of artificial paths obtained with the vanilla and supervised GAN. In both cases, the strong error outperforms the discrete-time schemes even at low values of Δt, suggesting that both GANs have learned a strong approximation. However, the vanilla GAN did not manage to find a strong approximation on all test problems (see e.g. Fig. 7)

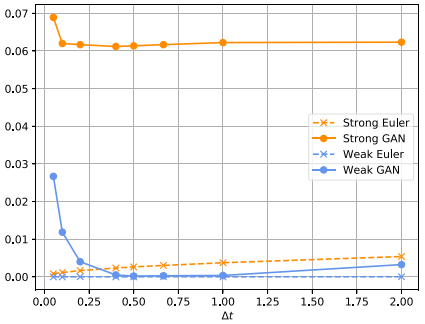
parameter $\bar{S}$ with increasing t at a rate defined by κ. We can test this property by sampling N values of $S_{t+\Delta t} \mid S_t$ repeatedly (e.g. 100 times) and taking the mean over all paths. The simulated process should converge in mean to $\bar{S}$. The result is shown in Fig. 4. The GAN indeed appears to revert to a mean, although it does not revert to the correct mean for each Δt. For lower values of Δt, the mean to which the GAN reverts is not equal to $\bar{S}$, which indicates that the distribution is captured less accurately than at higher values of Δt. Note that this experiment 'stress-tests' the iterative sampling method, as it is repeated 100 times, allowing errors to accumulate. In practice, one most likely only repeats the GAN output several times on the previous output. However, if many repeated samples are of interest, the architecture should be extended to include the possibilities for online corrections along the path.

4.3 Weak and strong error

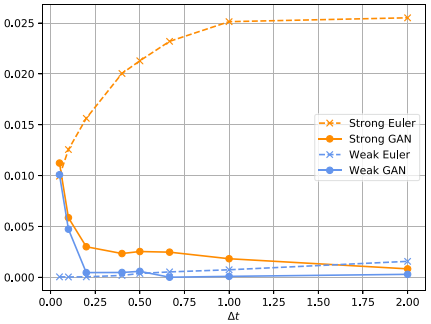
We iteratively sample from the process $\hat{S}_{t+\Delta t} \mid S_t$ with both the vanilla GAN and supervised GAN, on a discretisation $\{0, \Delta t, \ldots, N\Delta t\}$ with $\Delta t = \frac{T}{N}$ and $T = 2$. The



(a) Vanilla GAN path



(b) Vanilla GAN: e_w and e_s

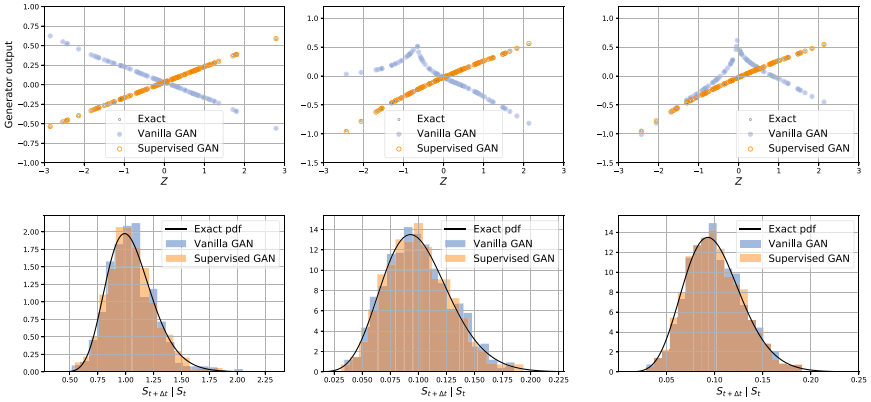


(c) Supervised GAN: e_w and e_s

Fig. 6 Example of failure of the vanilla GAN to provide a strong approximation on the CIR process if the Feller condition was satisfied. It fails to converge path-wise (**a**), which is reflected in the strong error (**b**). S_0 was set to 0.1

input to both GANs, $Z \sim N(0, 1)$, is stored at each time point and re-used for the Euler and Milstein approximation. Note that if we chose a different Z for the discrete-time schemes, we could not compare the results path-wise. The experiment is repeated for $N \in \{40, 20, 10, 5, 4, 3, 2, 1\}$ steps, yielding different choices of Δt. Using this setup, 100,000 paths were generated for each choice of N and the weak and strong error have been plotted versus Δt in Fig. 5.

When the Feller condition was not satisfied, the modified Milstein scheme did not perform better than the modified Euler scheme, which is why it was left out of this experiment. Both GANs yield a lower strong error even at small values of Δt across all three figures, which suggests that both GANs provide a strong approximation. For the vanilla GAN, this is a special case, as we see in Fig. 6, in which we provide an example if the Feller condition is satisfied. We study this phenomenon in more detail in the succeeding paragraph. The weak error in Fig. 5a, b on this problem is relatively high compared to the Euler scheme, which can be explained by the choice of parameters in this experiment. The mean reversion parameter $\bar{S}$ was set to 0.1, which is equal to S_0. This means that the Euler scheme starts at exactly the correct



(a) GBM **(b)** GANs from Figure 6a **(c)** Feller satisfied, second example

Fig. 7 Top row: the map $Z \mapsto G_\theta(Z, S_t, \Delta t)$ with $S_t = 0.1$ and $\Delta t = 1$ for three different examples. Each figure shows a scatter plot of the generator output of both GANs on the same 100 input samples Z. Bottom row: corresponding histograms of based on 1000 input samples Z

mean from time t_0. If we change S_0 to 0.01, the supervised GAN also outperforms the Euler scheme in weak error at approximately Δt greater than 0.5. The performance of both GANs is not uniform in Δt, which is particularly pronounced at low values. The opposite is true for the Euler scheme, which becomes increasingly accurate for decreasing Δt. Note that for an ideal GAN, the weak and strong error would not depend on Δt.

4.4 Map learned by vanilla and supervised GAN

We now test the reasoning in Sect. 2.4 empirically and study the map learned by both GAN architectures, i.e. the output $\left(S_{t+\Delta t} \mid S_t\right)(\omega)$ given an input $Z(\omega) \sim N(0,1)$ for an event $\omega \in \mathcal{F}_t$. Instead of $S_{t+\Delta t} \mid S_t$, we plot the output of the pre-processed data R_t on which the GANs were trained, i.e. logreturns for GBM and CIR with Feller condition violated, scaling with $\bar{S}$ if the Feller condition is violated. In Fig. 7, we show three different examples of the vanilla GAN failing to provide a strong approximation, although the approximation of the distribution is close.

Each of the examples in Fig. 7 gives rise to different pathological behaviour on the side of the vanilla GAN. The map on the left corresponds to 'mirrored' paths compared to the strong solution, corresponding to the weakly unique 'twin' solution to the strong solution with $Z \leftarrow -Z$, which is equal in distribution, but not path-wise. This is exactly the φ^- from Example 2.1. Note how the logreturns transformation makes the GBM problem trivial: the conditional GAN learns the slope and intercept of a straight line. In the centre and right figure, the vanilla GAN has not simply learnt a weakly unique solution with opposite sign, it has learned a map that gives rise to a similar distribution as the reference, but

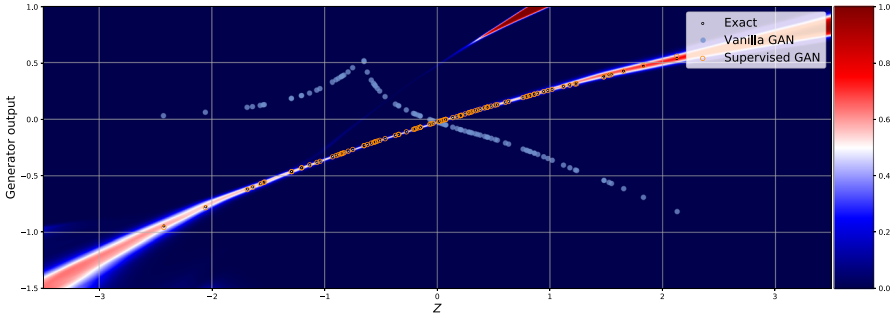


Fig. 8 Discriminator output corresponding to Figs. 6a and 7b output after 40,000 training steps for varying Z and $G_\theta(Z, S_t, \Delta t)$, with fixed $S_t = 0.1$ and $\Delta t = 1.0$. The discriminator identifies the region in which the exact samples lie for each combination of $(Z, \Delta t, S_t)$

corresponds to an entirely different map from Z to the GAN output. This would correspond to a map that generates samples within some ϵ from the target distribution, but where φ_θ itself is very different from φ^+ and φ^-, as we discussed in Eq. (14). This again led to the paths not being equal path-wise to the strong solution. The maps in the centre and rightmost figures are not bijective, since there are two returns for some inputs Z. Furthermore, the rightmost example shows that the vanilla GAN may be highly sensitive to small changes in the input. E.g. for Z around 0, the output can change very rapidly for a small perturbation in Z.

In all experiments performed in preparation for this work, the supervised GAN was able to provide a strong approximation, which is visible in Fig. 7 by the orange data points completely overlapping with the exact samples. The supervised GAN thus learns the map corresponding to the inverse function $F^{-1}_{S_{t+\Delta t}|S_t}(F_Z(Z))$.

4.5 Supervised GAN discriminator output

We can visualise how the supervised GAN learns by visualising the discriminator output and overlay the generator output. This way, we show explicitly how the discriminator scores each input sample. The generator is given by the function $G_\theta : \mathbb{R}^3 \to \mathbb{R}$ with input $(Z, S_t, \Delta t)$, while the discriminator is given by $D_\alpha : \mathbb{R}^4 \to [0, 1]$ with inputs $((S_{t+\Delta t} \mid S_t), Z, S_t, \Delta t)$. If we fix S_t and Δt, say at 0.1 and 1.0, respectively, we can visualise the discriminator output on [0, 1] with a colormap on the space $(Z, G_\theta(Z, S_t, \Delta t))$, which is shown in Fig. 8.

Upon convergence of the GAN, the discriminator output will be around 0.5 in a neighbourhood of the generated samples, as it is no longer able to distinguish between the reference and generated samples. This corresponds to the 'white band' in Fig. 8, which is exactly where the exact samples and supervised GAN samples can be found. If the vanilla GAN samples would have been provided to the supervised GAN discriminator, it would have classified all samples as fake,

as is visible in the figure by the fact that all the vanilla GAN data points lie in the dark blue region. This shows how the supervised GAN discriminator rules out any map other than the strong solution.

5 Discussion

Supervised learning One could argue that GANs are not needed to solve our problem, since the map $Z \mapsto \varphi_\theta(Z)$ could have equally been trained using only a 'generator' combined with an L_2 loss. This is possible since we had available the underlying map $F_Z^{-1} \circ F_{S_{t+\Delta t}|S_t}$ and were able to build a training set with examples of $((S_{t+\Delta t} \mid S_t), Z)$. However, using a supervised variant of the GAN as a reference model allowed us to compare both GAN-architectures directly, using the same learning algorithm.

Beyond GBM and the CIR process For general 1D Itô SDEs, where $F_{S_{t+\Delta t}|S_t}$ is not available analytically, one could use an empirical analogue instead, without any changes to the supervised GAN architecture. The only requirement is that the empirical approximation should be strictly increasing in order to find a unique Z for each data sample, which could be achieved e.g. with a non-decreasing interpolation scheme between the data points defining the ECDF. For higher dimensional SDEs, the prior input to the generator should be increased for each degree of freedom. If the Brownian motions are correlated, they can be written as a product covariance matrix and a vector independent Brownian motions, using Cholesky decomposition [9]. The covariance matrix would be a function of the correlation coefficients of each of the correlated Brownian motions. A conditional GAN could then be trained, with correlation parameters $\rho_1, \rho_2, \ldots$ as an additional conditional input.

Large time steps We showed that the supervised GAN is able to approximate the conditional distribution accurately for large time steps. 'Large' here meant large compared with a discrete-time approximation, which we used to benchmark our results. However, this may be considered unfair, since time steps of e.g. 1,2 are unrealistic for discrete-time schemes. On the other hand, the supervised GAN outperformed the discrete-time schemes on time steps below 1 as well, only struggling with the smallest of time steps. The comparison was sufficient to show that the supervised GAN is able to approximate the target SDE path-wise.

Data pre-processing On all benchmarks, performance of both GANs decreased the lower we chose Δt. This may seem counter-intuitive, as discrete-time schemes improve with decreasing Δt. However, since our model approximates an exact simulation scheme, the accuracy should theoretically not depend on Δt at all. The dependence of performance on Δt reflects the ability of the GAN to approximate the target distribution conditional on Δt. Neural networks tend to learn slower on input samples with lower variance [36]. This is because the gradient update for each weight scales with the variance of the input samples. Although the data were pre-processed by taking logreturns or scaling with $\bar{S}$, the *in-class* variance is still non-constant. The more conditional classes are added that affect the variance, the more

pronounced this result would be. An example would be if the parameter γ from the CIR process were added as an additional conditional input.

One way to counter the in-class variance would be to standardise each class individually. However, the post-processing step would then require knowledge of the mean and variance of the training set batches. A different route may be through scaling each training point with its corresponding Δt and S_0. However, in order to achieve unit variance, one would need very specific knowledge of the output distribution, which may be restrictive. Additionally, the heavy tails in the distributions make traditional standardisation techniques ineffective.

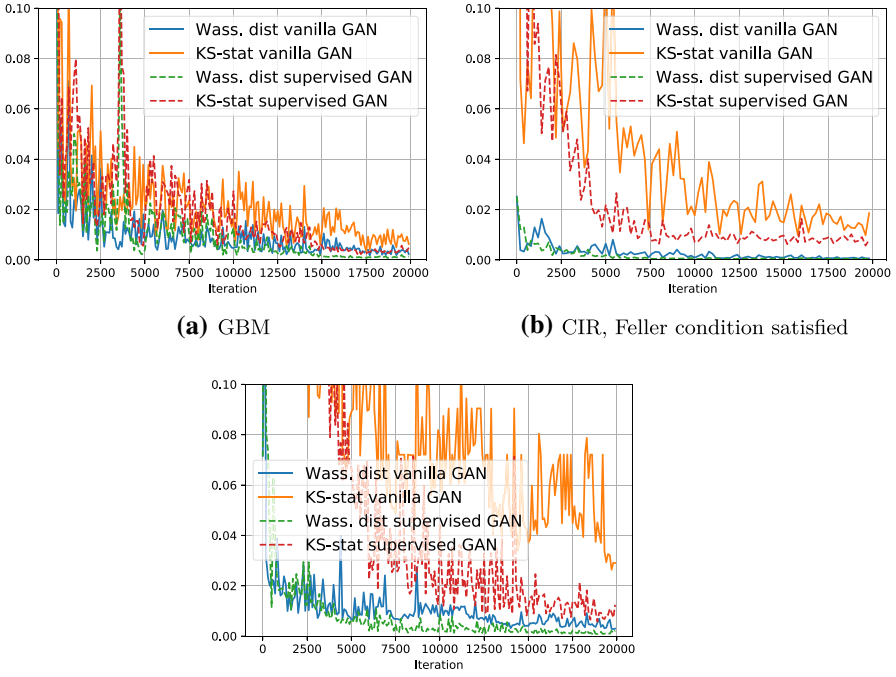
Full parameter range The conditional GAN architecture for modelling the conditional distribution could be further generalised to include the full parameter set of the SDE, allowing the GAN to learn an entire family of SDEs at once, as is done in [16] for the SCMC implementation. In this work, we developed a conditional GAN that was sufficient to demonstrate path-wise convergence. In future work, it would be interesting to test the GAN on the full parameter range of SDEs as well, if the challenge of pre-processing the data without incorporating knowledge of the target distribution could be resolved.

6 Conclusion and outlook

We proposed a GAN-based architecture for exact simulation of Itô SDEs. Specifically, we approximated the conditional probability distribution of 1D geometric Brownian motion (GBM) and the Cox-Ingersoll-Ross (CIR) process with a conditional GAN. The GAN was conditioned on the time interval length and the preceding value along the path and was used to construct artificial asset paths by iterative sampling from the conditional distribution. We argued that for unsupervised generative models based on divergence measures, there are no guarantees about the input-output map learned by the neural network. This is because the network parameters are varied only to minimise a quantity such as the Jensen-Shannon divergence, but no restriction is applied on the underlying map. We demonstrated experimentally how this could lead to non-unique and non-parsimonious input-output maps by the generator. In the context of SDEs, we showed how this implies that the vanilla GAN is unable to reliably provide a strong approximation. We replaced the vanilla GAN by a supervised GAN, which learns how a random input maps to the target variable explicitly. This supervised GAN was able to provide a strong approximation in all cases. Additionally, the approximation in distribution by the supervised GAN was more accurate under identical learning parameters and network capacity. We see two main directions for future work. Firstly, our findings motivate users of generative models to study the input-output map learned by the model explicitly and verify qualitative properties such as smoothness. This aligns well with efforts to constrain the generator, such as the 'potential flow generator' introduced in [44], that uses optimal transport to constrain the generator map. Secondly, our conditional GAN architecture could be further extended to include the SDE parameters as well, as is done for the 'Seven-League' collocation sampler in [16]. This would allow exact simulation of entire classes of SDEs instead of a specific choice of parameters. Since

Table 1 Network architectures

	Generator		Discriminator	
Optimiser	Adam		Adam	
	$lr = 10^{-4}, \beta_1 = 0.5, \beta_2 = 0.999$		$lr = 5 \times 10^{-4}, \beta_1 = 0.5, \beta_2 = 0.999$	
Layer	Size	Activation	Size	Activation
Input layer	$1+c$	LeakyReLU, negative slope=0.1	$1+c$	LeakyReLU, negative slope=0.1
Hidden layers 1-4	200	LeakyReLU, negative slope=0.1	200	LeakyReLU, negative slope=0.1
Output layer	1	None	1	Sigmoid



(a) GBM

(b) CIR, Feller condition satisfied

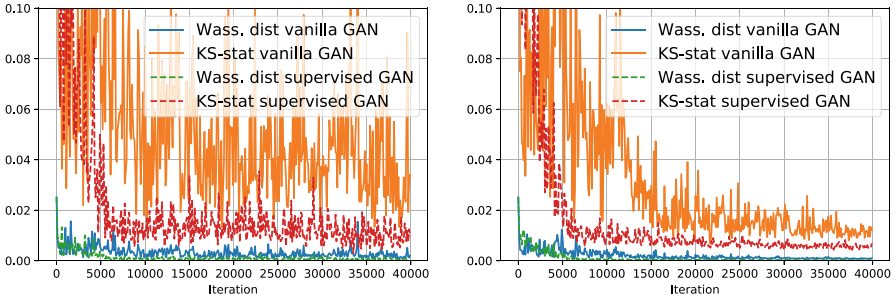
(c) CIR, Feller condition violated

Fig. 9 KS statistic and Wasserstein distance during training on a test set of 100,000 i.i.d. samples from the distribution of $S_{t+\Delta t} \mid S_t$. The default parameters listed in Sect. 1 were used to generate test sets. The supervised GAN converges faster than the vanilla GAN on both metrics

we showed how supervised learning can be used for Itô SDEs, the GAN architecture itself can be replaced by a single generator, trained on e.g. the mean-squared error. Extensions of our architecture, along with the methods we used for studying the output may be applied on more general problems, such as higher dimensional SDEs or non-Itô SDEs.

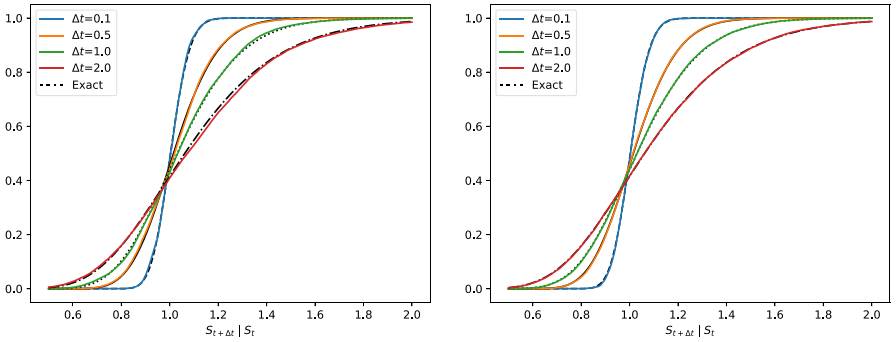
Network architectures

The architectures of the feed-forward neural networks of the generator and discriminator are shown in Table 1. c equals the amount of conditional parameters of the conditional GAN. $c = 1$ for GBM and $c = 2$ for the CIR process. If the discriminator is informed with Z, i.e. the supervised GAN, the discriminator input is further increased by 1 for the input Z. The batch size was set to 1000. Batches were sampled uniformly with replacement from a training set of 10^5 training samples. The GAN was trained for a fixed amount of 200 epochs. An additional learning rate schedule was created to stabilise GAN training. This schedule was used for the generator, where the learning rate was decreased by a factor 1.05 every $n_{\text{LR}} = 500$ iterations.



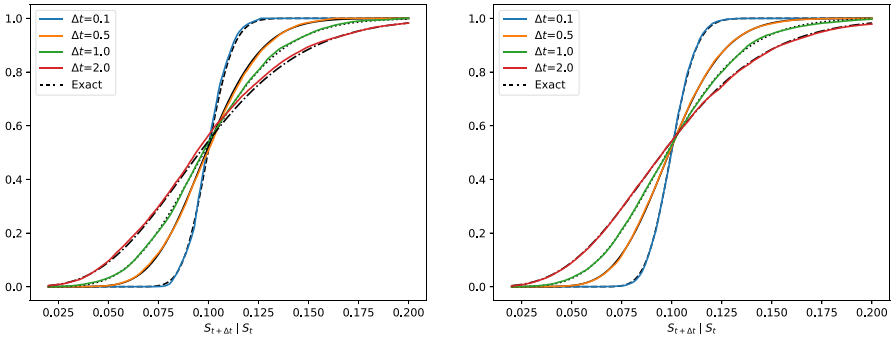
(a) No schedule **(b)** Divide by 1.05 every 500 iterations.

Fig. 10 Example of the effect of the learning rate schedule during training on the CIR process if the Feller condition is satisfied. The gradually decreasing learning rate allows the generator to more accurately converge to the target distribution



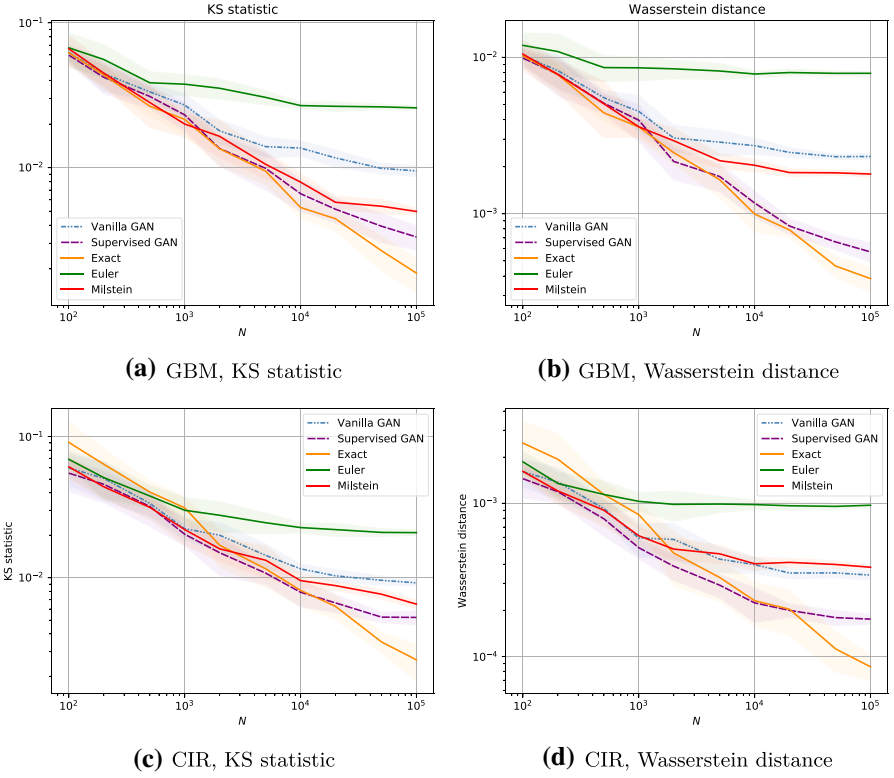
(a) Vanilla GAN **(b)** Supervised GAN

Fig. 11 ECDF plots of the conditional GAN output on the GBM problem for various choices of Δt. Note that pre-processing the data with logreturns removes dependence on S_t. In this example, $S_t = 1$



(a) Vanilla GAN **(b)** Supervised GAN

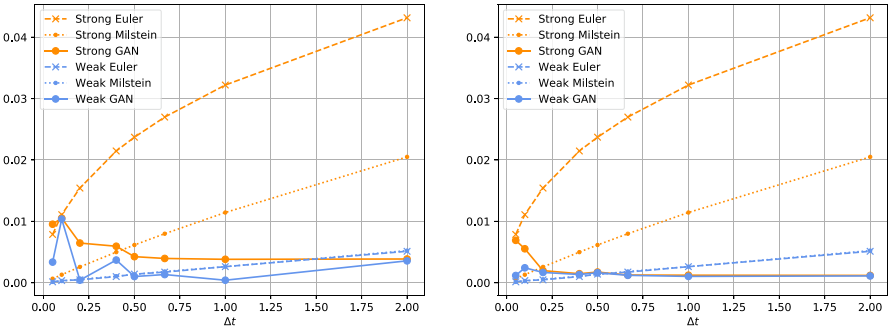
Fig. 12 ECDF plots of the conditional GAN output on the CIR process with the Feller condition satisfied, for various choices of Δt. S_t was held fixed at 0.1



(a) GBM, KS statistic **(b)** GBM, Wasserstein distance

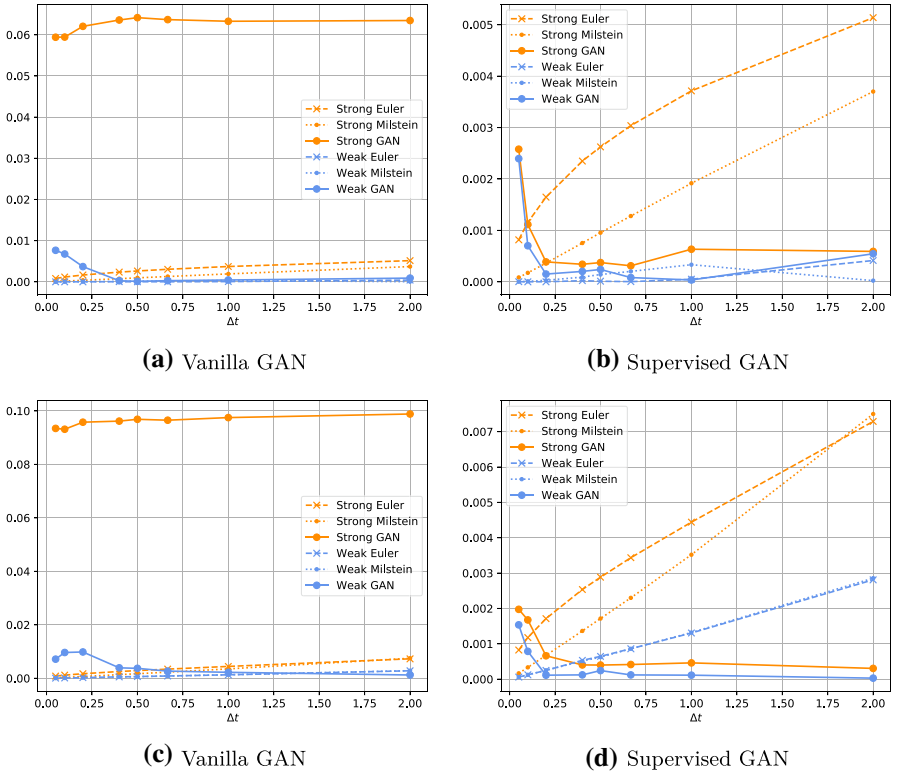
(c) CIR, KS statistic **(d)** CIR, Wasserstein distance

Fig. 13 KS statistic and Wasserstein distance at $\Delta t = 0.4$, versus the size of the test set. The confidence bands show the standard deviation based on 10 repetitions of the experiment, i.e. 10 iid samples of N random inputs to both GANs. The mean of both statistics is reported in the solid and dashed lines



(a) Vanilla GAN **(b)** Supervised GAN

Fig. 14 GBM: Weak and strong errors at time $T = 2$ of paths constructed using the vanilla GAN and supervised GAN, compared with the Euler scheme, Milstein scheme and exact scheme. $S_0 = 1$, $\Delta t = 0.05$, i.e. 40 steps between 0 and 2



(a) Vanilla GAN **(b)** Supervised GAN

(c) Vanilla GAN **(d)** Supervised GAN

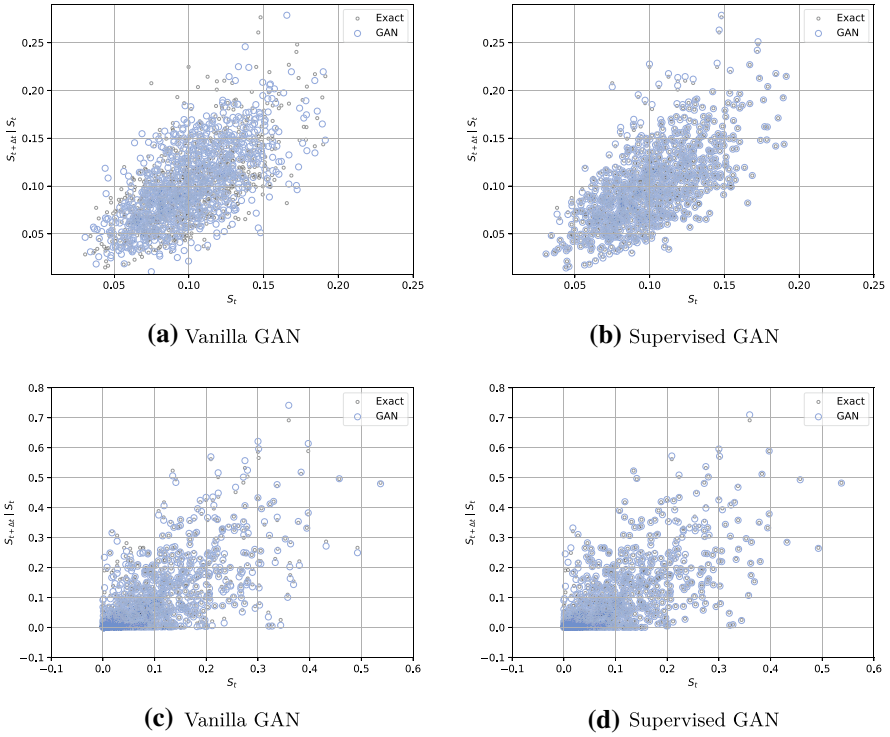
Fig. 15 CIR process with Feller condition satisfied: weak and strong errors at time $T = 2$ of paths constructed using the vanilla GAN and supervised GAN, compared with the Euler scheme, Milstein scheme and exact scheme. Δt ranges between 0.05 and 2. In (**a**, **b**), $S_0 = \bar{S}$, which gives the discrete-time schemes an advantage, as they have the mean reversion 'built-in'. Therefore, we also show the results if $S_0 \neq \bar{S}$, e.g. if $S_0 = 0.25$ in (**c**, **d**)

The learning rate for the discriminator was set to 5× that of the generator learning rate, which was found to lead to faster convergence in the first epochs and a better approximation of the optimal discriminator, cf. [29, section 2].

Saturating activation functions, such as a tanh or sigmoid, have also been considered. However, since the heavy left-tails of the target distributions persist after a preprocessing step (for the CIR process), the distribution of the hidden state will have a heavy tail as well. A saturating activation would then be undesirable, as it makes the tail less important in the saturating region. ReLU-type activations are not affected, as they are non-zero on $[0, \infty)$ and do not saturate.

KS statistic and Wasserstein distance during training

The KS statistic and Wasserstein distance were computed during training on a test set of 10^5 samples for GBM and the two instances of the CIR process. Figure 9 compares the training process of the vanilla and supervised GAN. Under equal training



(a) Vanilla GAN **(b)** Supervised GAN

(c) Vanilla GAN **(d)** Supervised GAN

Fig. 16 Scatter plot of $S_{t+\Delta t} \mid S_t$ versus S_t. In (**a**, **b**), the Feller condition is satisfied, while in (**c**, **d**) it is violated. In both cases, the similar shapes of the point clouds reveal that the autocorrelation structures of the GAN output and reference samples are similar, demonstrating good weak approximation capabilities for $\Delta t = 1$. Strong approximation is reflected in the data points overlapping with the exact samples

conditions, the supervised GAN converges faster and achieves a better approximation in distribution in all three cases. Figure 10 shows the effect of using a learning rate schedule for the generator (see Figs. 11, 12, 13, 14, 15 and 16, for more detailed output).

Results on GBM and the case Feller condition satisfied

Weak and strong error

Here, we show the weak and strong error obtained with artificial paths constructed with the vanilla and supervised GAN, this time for GBM and the CIR process if the Feller condition is satisfied. In the GBM example, the vanilla GAN happened to find a strong approximation, i.e. the same map as the conditional inverse distribution on the prior (Eq. (16)). On the example we show for the CIR process with the Feller condition satisfied, it did not manage to provide a strong approximation.

Details on the model parameters

CIR process SDE parameters In the case of the CIR process, the conditional random process $S_{t+\Delta t} \mid S_t$ follows a scaled non-central χ^2-distribution with some non-centrality parameter ξ, degrees of freedom δ and scaling factor $\bar{c}$ [9, 37]:

$$S_{t+\Delta t} \mid S_t \sim \bar{c}\, \chi^2(\xi, \delta), \tag{26}$$

where $\bar{c}, \xi$ and δ are related to the SDE parameters as shown in equations (27)–(29), cf. [37, p.392].

$$\xi(S_t, \Delta t) = \frac{4\kappa S_t e^{-\kappa \Delta t}}{\gamma^2\left(1 - e^{-\kappa \Delta t}\right)}, \tag{27}$$

$$\delta = \frac{4\kappa \bar{S}}{\gamma^2}, \tag{28}$$

$$\bar{c}(\Delta t) = \frac{\gamma^2}{4\kappa}\left(1 - e^{-\kappa \Delta t}\right). \tag{29}$$

Modified Euler and Milstein scheme for the CIR process A practical consideration for the CIR process is that discrete-time schemes could give rise to negative values, which are problematic when computing the square root in Eq. (24). Therefore, the Euler scheme will be replaced by what we will refer to as the (partially) *'truncated' Euler scheme*, as mentioned e.g. in [41]. In the case of the CIR process it is given by:

$$\hat{S}_{t+\Delta t} = \hat{S}_t + \kappa(\bar{S} - \hat{S}_t)\Delta t + \gamma \sqrt{\hat{S}_t^+}\sqrt{\Delta t} Z, \tag{30}$$

where $S_0 \in \mathbb{R}$ and $\hat{S}_t^+ := \max(\hat{S}_t, 0)$. $Z \sim N(0, 1)$. Note that the truncated Euler scheme may still produce negative paths, in which case the term with the Brownian motion is equal to zero at step $t + \Delta t$.

A modified version of the Milstein scheme can be defined as well. In [42], such a Milstein-type scheme has been proposed specifically for the CIR process, which we implemented as a reference to the CIR process. This *truncated Milstein scheme* is given by in Eq. (31).

$$\begin{aligned} \hat{S}_{t_{k+1}} = &\left(\left(\max\left(\frac{1}{2}\gamma\sqrt{\Delta t}, \sqrt{\max\left(\frac{1}{2}\gamma\sqrt{\Delta t}, \hat{S}_{t_k}\right)} + \frac{1}{2}\gamma\sqrt{\Delta t} Z_k\right)\right)^2\right. \\ &\left. + \left(\kappa\bar{S} - \frac{1}{4}\gamma^2 - \kappa\hat{S}_{t_k}\right)\Delta t\right)^+, \end{aligned} \tag{31}$$

where $\hat{S}_{t_0} = S_{t_0}$ and $(\cdot)^+ := \max(\cdot, 0)$. The one-step order of convergence of this scheme depends on the previous value S_{t_k}, time step Δt and degrees of freedom parameter δ [42]. However, the authors of [42] show that the scheme converges in L^p

with order $\frac{1}{2p}\min(1,\delta)$. Note that we could have used a Milstein scheme analogous to the truncated Euler scheme, but this led to inferior weak and strong errors compared to the scheme in [42].

Default training/testing parameters For GBM, the default parameters were $\mu = 0.05$ and $\sigma = 0.2$. The CIR parameters were $\bar{S} = 0.1, S_t = 0.1, \kappa = 0.1, \gamma \in \{0.1, 0.3\}$, where $\gamma = 0.1$ corresponds to the case where the Feller condition is satisfied ($\delta = 4$) and $\gamma = 0.3$ to the case where the Feller condition is violated ($\delta = 0.44$).

Conditional GAN training Both GANs were trained on a range of parameters of Δt and (for the CIR process) the 'previous value' S_t. It was found that choosing a discrete range parameters that recur many times outperformed a continuous range of unique parameters. For example, for Δt, a fixed list was created of times of interest $\Delta t \in \{0.05, 0.1, 0.2, 0.4, 0.5, 0.67, 1, 2\}$, each of which occured with equal frequency in a dataset of N_{train} training samples. This outperformed a continuous range of N_{train} time steps on [0, 2]. If more than one conditional parameter is chosen, a training set must be defined as the Cartesian product of the two discrete sets of training parameters. This was achieved by randomly permuting each vector of training samples and concatenating the results. This created ordered vectors of the pairs $(S_t, \Delta t)$, which were provided as input to a function that draws samples from the exact distribution of $S_{t+\Delta t} \mid S_t$ and provides corresponding standard normal variates Z to train the supervised GAN.

Autocorrelation structure for results on CIR process

To test whether the GAN has correctly captured the dependence of the conditional distribution on the previous S_t, we show a plot of $S_{t+\Delta t} \mid S_t$ versus S_t. These plots were made by first drawing 1, 000 samples using the exact distribution of $S_t \mid S_0$ at $t = 1$ and $S_0 = 0.1$. Then, an additional 1000 samples $S_{t+\Delta t} \mid S_t$ were drawn from the exact distribution conditional on the samples S_t, with $\Delta t = 1$. Z was computed using Eq. (17) and provided as input to the generator.

Declarations

Conflict of interest the authors declare that they have no conflict of interest.

References

1. Ian., G, Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K.Q. (eds.) Advances in neural information processing systems 27 (NIPS 2014). Curran, Red Hook, NY (2015)
2. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. In: 4th International Conference on Learning Representations, ICLR 2016—Conference Track Proceedings (2015). arXiv:1511.06434
3. Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., Lee, H.: Generative adversarial text to image synthesis. In: 33rd International Conference on Machine Learning, ICML 2016, vol. 48, pp. 1060–1069 (2016)
4. Zhu, J.-Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: The IEEE International Conference on Computer Vision (ICCV), pp 2242–2251 (2017). https://doi.org/10.1109/ICCV.2017.244
5. Arjovsky, M., Bottou, L.: Towards principled methods for training generative adversarial networks. In: Proceedings International Conference on Learning Representations (2017). https://openreview.net/forum?id=Hk4_qw5xe
6. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: International Conference on Machine Learning, vol. 70, pp 214–223. PMLR (2017)
7. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training GANs. In: Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29 (NIPS 2016). Curran, Red Hook, NY (2017)
8. Heusel, M., Ramsauer, H., Unterthiner, T, Nessler, B., Hochreiter, S.: GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In: Guyon, I., Von Luxburg, U., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30 (NIPS 2017). Curran, Red Hook, NY (2018)
9. Oosterlee, C.W., Grzelak, L.A.: Mathematical Modeling and Computation in Finance. World Scientific, Singapore (2019)
10. Berner, J., Grohs, P., Jentzen, A.: Analysis of the generalization error: empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations. SIAM J Math Data Sci **2**(3), 631–657 (2020)
11. Grohs, P., Hornung, F., Jentzen, A., Von Wurstemberger, P.: A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations. arXiv preprint arXiv:1809.02362 (2018)
12. Oksendal, B.: Stochastic Differential Equations: an Introduction with Applications. Springer Science & Business Media, Berlin (2013)
13. Kloeden, P.E., Platen, E., Schurz, H.: Numerical Solution of SDE Through Computer Experiments. Springer Science & Business Media, Berlin (2012)
14. Broadie, M., Kaya, Ö.: Exact simulation of stochastic volatility and other affine jump diffusion processes. Oper. Res. **54**(2), 217–231 (2006)
15. Grzelak, L.A.: The collocating local volatility framework—a fresh look at efficient pricing with smile. Int. J. Comput. Math. **96**(11), 2209–2228 (2019)
16. Liu, S., Grzelak, L.A., Oosterlee, C.W.: The Seven-League Scheme: Deep learning for large time step Monte Carlo simulations of stochastic differential equations. Risks **10**(3), 47 (2022)
17. Yang, L., Zhang, D., Karniadakis, G.E.M.: Physics-informed generative adversarial networks for stochastic differential equations. SIAM J. Sci. Comput. **42**(1), A292–A317 (2020)
18. Xie, Y., Franz, E., Chu, M., Thuerey, N.: Tempogan: a temporally coherent, volumetric GAN for super-resolution fluid flow. ACM Trans. Graph. (TOG) **37**(4), 1–15 (2018)
19. Joshi, A., Shah, V., Ghosal, S., Pokuri, B., Sarkar, S., Ganapathysubramanian, B., Hegde, C.: Generative models for solving nonlinear partial differential equations. In: Workshop at 33rd Conference on Neural Information Processing Systems (NeurIPS) (2019). https://ml4physicalsciences.github.io/2019/files/NeurIPS_ML4PS_2019_118.pdf
20. Abbati, G., Wenk, P., Osborne, M.A., Krause, A., Schölkopf, B., Bauer, S.: ARES and MarS—adversarial and MMD-minimizing regression for SDEs. In: 36th International Conference on Machine Learning, ICML 2019, vol. 97, pp. 1–10 (2019)

21. Chen, R.T.Q., Rubanova, Y., Bettencourt, J., Duvenaud, D.: Neural ordinary differential equations. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 31 (NeurIPS 2018). Curran, Red Hook, NY (2019)
22. Kidger, P., Foster, J., Li, X., Lyons, T.J.: Neural SDEs as infinite-dimensional GANs. In: International Conference on Machine Learning, Vol. 139, pp 5453–5463. PMLR (2021)
23. Wiese, M., Knobloch, R., Korn, R., Kretschmer, P.: Quant GANs: deep generation of financial time series. Quant. Finance **20**(9), 1419–1440 (2020)
24. Ni, H., Szpruch, L., Sabate-Vidales, M., Xiao, B., Wiese, M., & Liao, S.: Sig-Wasserstein GANs for time series generation. In: Proceedings of the Second ACM International Conference on AI in Finance, pp. 1–8. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3490354.3494393
25. Fu, R., Chen, J., Zeng, S., Zhuang, Y., Sudjianto, A.: Time series simulation by conditional Generative Adversarial Net. arXiv preprint arXiv:1904.11419 (2019)
26. Cuchiero, C., Khosrawi, W., Teichmann, J.: A generative adversarial network approach to calibration of local stochastic volatility models. Risks **8**(4), 101 (2020)
27. Klenke, A.: Probability Theory. Universitext. Springer-Verlag London Ltd., London (2008)
28. Peters, H.: Game Theory: A Multi-Leveled Approach. Springer, Berlin (2015)
29. Biau, G., Cadre, B., Sangnier, M., Tanielian, U., et al.: Some theoretical properties of GANs. Ann. Stat. **48**(3), 1539–1566 (2020)
30. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014)
31. Lin, J.: Divergence measures based on the Shannon entropy. IEEE Trans. Inf. Theory **37**(1), 145–151 (1991)
32. Kullback, S., Leibler, R.A.: On information and sufficiency. Ann. Math. Stat. **22**(1), 79–86 (1951)
33. Bottou, L.: Large-scale machine learning with stochastic gradient descent. In: Lechevallier, Y., Saporta, G. (eds.) Proceedings of COMPSTAT'2010, pp. 177–186. Springer, Heidelberg (2010)
34. Simard, R., L'Ecuyer, P., et al.: Computing the two-sided Kolmogorov-Smirnov distribution. J. Stat. Softw. **39**(11), 1–18 (2011)
35. Ramdas, A., Trillos, N.G., Cuturi, M.: On Wasserstein two-sample testing and related families of nonparametric tests. Entropy **19**(2), 47 (2017)
36. LeCun, Y., Bottou, L., Orr, G.B., Müller, K.-R., et al.: Neural networks: Tricks of the trade. Springer Lect. Notes Comput. Sci. **1524**(5–50), 6 (1998)
37. Cox, J.C., Ingersoll Jr, J.E., Ross, S.A.: A theory of the term structure of interest rates. In: Bhattacharya, S., Constantinides, G.M. (eds.) Theory of valuation, pp. 129–164. World Scientific (2005)
38. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: Proc. icml **30**, 3 (2013)
39. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L, et al.: Pytorch: an imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32 (NeurIPS 2019). Curran, Red Hook, NY (2020)
40. Le Gall, J.-F.: Brownian Motion, Martingales, and Stochastic Calculus, vol. 274. Springer, Berlin (2016)
41. Labbé, C., Rémillard, B., Renaud, J.F.: A simple discretization scheme for nonnegative diffusion processes with applications to option pricing. J. Comput. Finance **15**(2), 3–35 (2011)
42. Hefter, M., Herzwurm, A.: Strong convergence rates for Cox-Ingersoll-Ross processes-full parameter range. J. Math. Anal. Appl. **459**(2), 1079–1101 (2018)
43. Albrecher, H., Mayer, P., Schoutens, W., Tistaert, J.: The little Heston trap. Wilmott **1**, 83–92 (2007)
44. Yang, L., Karniadakis, G.E.: Potential flow generator with L2 optimal transport regularity for generative models. IEEE Trans. Neural Netw. Learn. Syst. **33**(2), 528–538 (2020). https://doi.org/10.1109/TNNLS.2020.3028042

Authors and Affiliations

Jorino van Rhijn[1] · Cornelis W. Oosterlee[2] · Lech A. Grzelak[3,4] · Shuaiqiang Liu[1]

Jorino van Rhijn
jorinovrhijn@hotmail.com

1 Centrum Wiskunde en Informatica (CWI), Amsterdam, The Netherlands

2 Mathematical Institute, Utrecht University, Utrecht, The Netherlands

3 Delft University of Technology, Delft, The Netherlands

4 Rabobank, Utrecht, The Netherlands