# Multivariate Time Series Modelling with Neural SDE driven by Jump Diffusion

Kirill Zakharov[1][0000−0001−5774−4076]

Research Center "Strong Artificial Intelligence in Industry", ITMO University, Saint Petersburg, 199034, Russia
kazakharov@itmo.ru

**Abstract.** Neural stochastic differential equations (neural SDEs) are effective for modelling complex dynamics in time series data, especially random behavior. We introduced JDFlow, a novel normalizing flow method to capture multivariate structures in time series data. The framework involves a latent process driven by a neural SDE based on the Merton jump diffusion model. By using maximum likelihood estimation to determine the intensity parameter of the Poisson process in neural SDE, we achieved better results in generating time series data compared to previous methods. We also proposed a new approach to assess synthetic time series quality using a Wasserstein-based similarity measure, which compares signature cross-section distributions of original and generated time series.

**Keywords:** neural stochastic differential equations · normalising flows · Merton jump diffusion · multivariate time series modelling · path signature.

## 1 Introduction

In the ever-evolving landscape of data science and machine learning, the field of time series modeling has emerged as an essential and challenging area of research. Time series data, with its unique temporal dependencies and sequential patterns, finds applications in various fields such as finance, healthcare, and climate science, among others [1,2,3]. The accurate modeling of time series is crucial for creating robust models and understanding complex systems. One approach to modelling time series is through generative models [4], which have practical applications in anomaly detection [5] and data augmentation [6]. In this paper, we propose a novel method based on normalizing flows and neural SDEs for time series generation and modelling. Especially, we aim to create a model that can account the jumps in real markets, utilizing the Merton model [3] as the jump framework.

Normalising flows are a family of generative models with tractable density estimation. The main idea is to transform the initial complicated data distribution to a simple one, by composition of several functions $f_i$. There are some

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_26

constraints to implement such architecture: i) each function $f_i$ should be a bijection (should exist an invertible transformation $g_i^{-1} = f_i$); ii) they should have an analytically closed form and must be differentiable; iii) the Jacobian determinant of $f_i$ should be easily calculable. The composition of the functions $f_i$ is called the *flow*. Functions $g_i = f_i^{-1}$ form the *forward* or *generative direction* and $f_i$ form the *backward* or *normalising direction* [4].

Stochastic differential equations (SDEs) are applied in various fields due to their ability to model systems, influenced by both deterministic and random factors. Here are some common applications: (i) in finance, SDEs determine the asset price dynamic, e.g., the famous Black-Scholes model for option pricing involves SDEs, by using geometric Brownian motion to determine stock prices; (ii) in physics, SDEs are used to model a system with random fluctuations. For example, the motion of particles in a fluid can be described by Langevin equations; (iii) SDEs are employed in modelling biological systems, e.g., population dynamics and the spread of diseases can be described using stochastic models.

In this article, we consider the Itô's type of *stochastic differential equations* [7] with additional jump component, which defined as (1).

## 2 Method

Figure 1 shows the general method's pipeline. The red section is dedicated to the latent process driven by neural SDE from the Section 2.1. The blue section is devoted to intensity estimation from the Section 2.2. Finally, the general generative model framework is discussed in the Section 2.3.

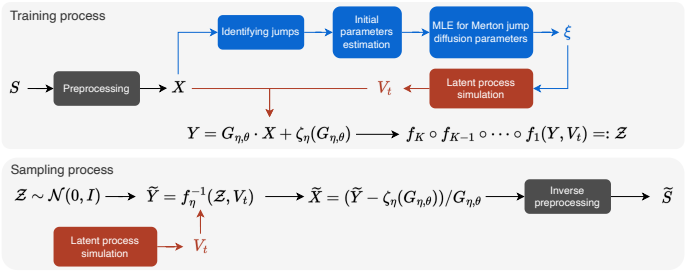



Fig. 1: General pipeline.

### 2.1 Latent process and neural SDE formulation

Consider the time series $S$ of the length $T$ with dimension $M$. For a given $S$, we apply the preprocessing procedure, which scales the series in the range $[0, 1]$ and result it as $X \in \mathbb{R}^{M \times T}$. The main idea of our approach is to use the jump diffusion dynamic instead of the standard Itô's diffusion. To add the jump diffusion, we implement a *latent process* $V_t$. We suppose, that $V_t$ is square-integrable and it follows the jump diffusion dynamic (neural SDE [8]),

$$dV_t = \mu_\theta(V_t, t)dt + \sigma_\theta(V_t, t)dW_t + J_\theta(V_t, t)dP_t, \tag{1}$$

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_26

where $W_t$ is the Wiener process, $P_t$ is the Poisson process, $\mu_\theta$, $\sigma_\theta$, $J_\theta$ are the drift, diffusion, and jump magnitude parameters, respectively. $W_t$ and $P_t$ are assumed to be independent. The functions $\mu, \sigma, J$ are parameterised by neural networks instead of using the predefined dynamic. The networks' parameters are stored in $\theta$ for short. The networks $\mu, \sigma, J$ take the vector $V_t \in \mathbb{R}^M$ and the vector $(t, \sin t, \cos t) \in \mathbb{R}^3$ as input, where time component is augmented by sine and cosine transformations, and produce the vector of the dimension $M$. For hidden layers we use fully connected layers, with hidden dimension $2^7$, and hyperbolic function, as activation function.

The initial value $V_0$ of the process $V_t$ is identified by another neural network $\varphi_\theta$, which depends on the random noise $Z_t \in \mathbb{R}^{1\times M}$ and some initial information $X_0$ about $X$ (we use $X$'s values at zero time point $X_0 \in \mathbb{R}^{M\times 1}$). The network $\varphi_\theta$ takes the vector $(Z_t, X_0^\top) \in \mathbb{R}^{2M}$ as input, and its outcome is the initial value $V_0 \in \mathbb{R}^M$.

It is complicated to find an analytic solution of the equation (1), because the functions $\mu, \sigma, J$ have complicated structure (they are the neural networks with huge amounts of parameters). But, it is possible to use the discrete scheme to approximate the solution. We choose the standard one so-called the *Euler-Maruyama scheme* [9]. Finally, we can represent the equation (1) as the following discrete scheme,

$$V_{t_{i+1}} = V_{t_i} + \mu_\theta(V_{t_i}, t_i)\Delta t + \sigma_\theta(V_{t_i}, t_i)\varepsilon\sqrt{\Delta t} + J_\theta(V_{t_i}, t_i)P_{t_i}, \tag{2}$$

where $\Delta t = \frac{T}{\tau}$, $\tau$ is the number of steps in the discrete scheme, $P_{t_i}$ is the Poisson random variable with intensity $\xi\Delta t$, and $\varepsilon \sim \mathcal{N}(0,1)$. We denote the solution block with discrete Euler-Maruyama scheme as $F_\theta(V_0, \Delta t, \mu_\theta, \sigma_\theta, J_\theta)$, which will produce the multivariate time series of the dimension $M$ and the length $\tau$. We refer $\tau$ as hyperparameter and recommend to choose it not greater than $T$.

### 2.2 Intensity estimation with Merton model

Consider the log return process $R_t = \log \frac{X_{t+dt}}{X_t}$. If there is a jump in the time interval $[t_i, t_{i+1})$, then we can observe it by $R_{t_i}$. We separate the dynamic of $R_t$ in two components $R^J$ and $R^D$, using the hyperparameter $\lambda$ (threshold of a jump presence in $R_t$). The $R_J$ is dedicated to observations with jumps and $R_D$ with the diffusion part. We use the Merton model to represent the jump diffusion process (MJD), because it is more natural to use the Gaussian distribution for jump amplitude in real markets.

To get the intensity parameter $\xi$, we use the maximum likelihood approach for MJD on the given data. For that, we define the initial parameters to achieve better convergence as in [3].

$$\hat{\mu}_D = \frac{2\mathbb{E}[R^D] + \mathbb{V}ar[R^D]dt}{2dt},\ \hat{\sigma}_D^2 = \frac{\mathbb{V}ar[R^D]}{dt},\ \hat{\mu}_J = \mathbb{E}[R^J] - \left(\hat{\mu}_D - \frac{\hat{\sigma}_D^2}{2}\right)dt,$$
$$\hat{\sigma}_J^2 = \mathbb{V}ar[R^J] - \hat{\sigma}_D^2 dt,\ \ \hat{\xi} = \frac{\#R^J}{T}, \tag{3}$$

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_26

where $\#R^J$ is the number of jumps (cardinality of a set $R^J$).

Finally, we maximise the likelihood function for the log return process,

$$\log L(R|\xi, \mu_D, \sigma_D^2, \mu_J, \sigma_J^2) = \\ = \sum_{i=1}^{T} \log \left( \sum_{k=0}^{\infty} \frac{(\xi dt)^k}{k!} e^{-\xi dt} \cdot \mathcal{N}\left( R_i \middle| (\mu_D - \frac{\sigma_D^2}{2})dt + \mu_J k,\ \sigma_D^2 dt + \sigma_J^2 k \right) \right), \quad (4)$$

where $\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}$. The bounds for the optimisation task are $\xi \geq 0, \mu_D, \mu_J \in \mathbb{R}$, and $\sigma_D^2, \sigma_J^2 > 0$. The parameter $k$ is dedicated to a number of jumps in a single time period and it can be chosen empirically.

### 2.3 General framework

For the given $V_0, \Delta t$, the solution for equation (1) is equal to $V_t \equiv V_t^\theta = F_\theta(V_0, \Delta t, \mu_\theta, \sigma_\theta, J_\theta) \in \mathbb{R}^{\tau \times M}$. After that, we compute the non-linearity term as $Y \equiv Y_{\eta,\theta} = G_{\eta,\theta} \cdot X + \zeta_\eta(G_{\eta,\theta})$, where $G_{\eta,\theta} = e^{-\psi_\eta(V_t^\top)} \in \mathbb{R}^{M \times T}$. The neural networks $\zeta_\eta$ and $\psi_\eta$ are dedicated to the same part of the model and its parameters are stored in $\eta$. The network $\psi_\eta$ for each $m \in \{1, \ldots, M\}$ takes the vector from $\mathbb{R}^\tau$ as input and produces the vector from $\mathbb{R}^T$. Further, we identify the normalising flow for $Y_{\eta,\theta}$ as a composition $f_\eta(Y, V_t) = f_K \circ f_{K-1} \circ \cdots \circ f_1(Y, V_t)$ and the generative flow as $\widetilde{Y} \equiv f_\eta^{-1}(\mathcal{Z}, V_t) = f_1^{-1} \circ \cdots \circ f_{K-1}^{-1} \circ f_K^{-1}(\mathcal{Z}, V_t)$. The value $f_\eta(Y, V_t)$ should be approximately equals to $\mathcal{Z} \sim \mathcal{N}(0, I)$ by distribution.

We propose a new bijection function, which depends on the latent process $V_t$. The forward path of $f_i(Y, V_t)$ starts from separation of the process $Y \in \mathbb{R}^{M \times T}$ in two parts $Y_1 \in \mathbb{R}^{M \times d}$ and $Y_2 \in \mathbb{R}^{M \times (T-d)}$. Further, we calculate the term which depends on the latent process $\hat{p} = p_\eta(V_t^\top)$ to extract the information given on the latent dynamic. Then we implement the affine coupling layer [4] with following modifications,

$$\mathcal{Z}_1 = Y_1 + \hat{p}, \\ \mathcal{Z}_2 = \Lambda_\eta(\mathcal{Z}_1) \odot g_\eta(\nu_\eta(\mathcal{Z}_1^\top)^\top) + Y_2 \odot e^{\Sigma_\eta(\mathcal{Z}_1)}, \quad (5)$$

where $\odot$ is an element-wise product. After that, we concatenate the outcomes as $\mathcal{Z} = (\mathcal{Z}_1, \mathcal{Z}_2) \in \mathbb{R}^{M \times T}$. The networks $\Lambda_\eta, \Sigma_\eta$ have the same structure with one recurrent layer and several linear layers with sigmoid activation between them. Due to we use the affine coupling layer, we also permute the component $\mathcal{Z}_1$ with $\mathcal{Z}_2$ after each flow.

Our novel idea is to use the transpose version of $\mathcal{Z}_1$ to extract the information from the multivariate dynamic, utilising the neural network $\nu_\eta$, and we transpose the result again to get the term structure dynamic. Then we apply the neural network $g_\eta$ to extract once again the time structure component, but in a new representation of $\mathcal{Z}_1$. The networks $\nu_\eta, g_\eta$ were constructed with two linear layers with sigmoid activation function between them. The flow constructed in this manner maintains the form of the Jacobian of a coupling flow.

Therefore, the log determinant of Jacobian is equal to the sum of elements $\Sigma_\eta(\mathcal{Z}_1)(j)$, where $j \in \{d+1, \ldots, T\}$. For the likelihood distribution we choose the standard Gaussian one.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_26

We define the inverse flow as

$$\begin{aligned}\widetilde{Y}_2 &= (\mathcal{Z}_2 - \Lambda_\eta(\mathcal{Z}_1) \odot g_\eta(\nu_\eta(\mathcal{Z}_1^\top)^\top)) \odot e^{-\Sigma_\eta(\mathcal{Z}_1)},\\ \widetilde{Y}_1 &= \mathcal{Z}_1 - \hat{p}.\end{aligned} \tag{6}$$

The final value $\widetilde{Y}$ achieved by concatenating components $(\widetilde{Y}_1, \widetilde{Y}_2)$. After the flow we implement the inverse procedure for non-linearity term as $\widetilde{X} = (\widetilde{Y} - \zeta_\eta(G_{\eta,\theta}))/e^{-\psi_\eta(V_\theta^\top)}$ by sampling the new latent process with discrete scheme (2). Finally, we get a synthetic multivariate time series $\widetilde{S}$, by applying the inverse preprocessing procedure to $\widetilde{X}$.

## 3 Experimental study

### 3.1 Data description

To evaluate the proposed model we use three multivariate time series. The first is dedicated to the jump diffusion so-called Merton model which driven by SDE in a risk-neutral measure as

$$dX_t = \left(r - \xi \cdot (e^{\mu_J + 0.5\sigma_J^2} - 1) - \frac{1}{2}\sigma_D^2\right)dt + \sigma_D dW_t + J dP_t. \tag{7}$$

As parameters we use $\xi = 15, \mu_J = 0, \sigma_J = 0.2, r = 0.04, \sigma_D = 0.6$. The length of the time series and the dimension are equal to 1000 and 10, respectively. By the Merton jump diffusion, we check the ability of the models to detect the complicated dynamics.

The second time series is the diffusion process, driven by DCL stochastic process [10], with $\theta = 1$ and $\delta = 2$. The time series length is equal to 500 and the dimension is 5. Using the DCL process, we wish to test the smoothness of the resulting series.

The third time series is derived from the kaggle stock dataset[1] to analyse the ability of the models to generate the realistic series for the market purposes. The multivariate attributes are formed by open, close, low, and high prices with 2645 total observations.

### 3.2 Quality assessment

We employ three methods to compare results: Fourier Flow [11], fSDE [12] and PAR from SDV [13].

We choose the Jensen-Shannon divergence and $W_1$-distance to compare the distribution similarity. We also utilise the forecasting model to assess the prediction error, $MSE$. We use the local extrema QQ-plot, to evaluate the ability of the model to represent the local dynamic features [14]. To evaluate the multivariate similarity, we use t-SNE plots [15], which provides the dimension reduction with structure preserving.

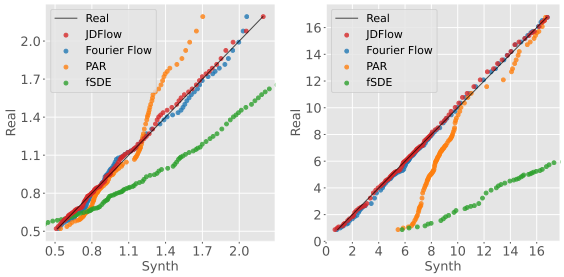
[1] https://www.kaggle.com/datasets/borismarjanovic/price-volume-data-for-all-us-stocks-etfs

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_26

We introduced a new method to assess global dynamic similarity by comparing signature distributions at different time points. By calculating path integrals and extracting cross sections over the interval [0, T], we can compare the distributions of real-world time series signatures with synthetic data. Using the $W_1$-distance metric, we visualize the differences in a plot for all models. The path integrals in the signatures play a key role in determining the time series dynamics uniquely [16]. This approach helps identify specific time periods where the synthetic data deviates from the real data.

### 3.3 Results

Figure 2 illustrates QQ-plots for local extrema. Our approach JDFlow (red color) is better to approximate the local structure, by matching the diagonal line, which refers to a real extrema quantile (black color). Fourier Flow is also close to the real time series, but PAR (orange color) and fSDE (green color) are too far from the diagonal. For example, PAR's synthetics differ in the first quantiles in the stock dataset, which means it has greater local minima values, than it has the initial time series.




(a) Merton jump diffusion (b) Stock

Fig. 2: QQ-plot for the local extrema comparison.

Figure 3 shows the $W_1$-distance between cross sections of the signatures in different time periods. The best values should be close to a zero line (black color). PAR and fSDE models greatly differ from the real time series, especially in comparison with our approach JDFlow and Fourier Flow. For the Merton jump diffusion (Fig. 3a), the PAR and fSDE models have a $W_1$-distance that increases significantly over time. This means that their synthetics branch out too much over time. The same structure is observed in the stock data, only JDFlow and Fourier Flow accurately discover the initial data patterns.

Figure 4 shows distinctions in multivariate structure. As a result, real-world time series, JDFlow, and Fourier Flow synthetics have similar multivariate structure. fSDE model repeats the local patterns of the source data, but globally behaves differently. The PAR synthesizer doesn't match the initial multivariate dynamic.

The quantitative evaluation can be found in the Table 1. For all time series JDFlow has shown the best results in terms of distribution similarity (blue color

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_26


(a) Merton jump diffusion (b) Stock

Fig. 3: Signature cross section distribution comparison.

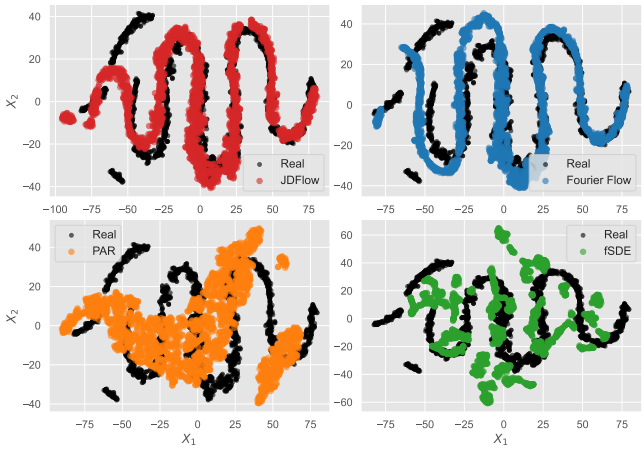



Fig. 4: t-SNE plot for stock.

for best results). When analysing $MSE$ metric for the forecasting task, JDFlow shows the best results and even decrease the initial time series prediction error.

Table 1: Quality assessment with quantitative metrics.

| Data | Models | $W_1$ | $D_{JS}$ | $MSE$ |
|---|---|---|---|---|
| **Merton jump diffusion** | Real | 0.000 | 0.0000 | 0.059 |
| | JDFlow | 0.003 | 0.005 | 0.057 |
| | Fourier Flow | 0.153 | 0.136 | 0.439 |
| | PAR | 0.093 | 0.078 | 0.288 |
| | fSDE | 2.275 | 0.303 | 2.181 |
| **DCL process** | Real | 0.000 | 0.000 | 0.019 |
| | JDFlow | 0.002 | 0.019 | 0.019 |
| | Fourier Flow | 0.094 | 0.345 | 0.101 |
| | PAR | 0.041 | 0.218 | 0.030 |
| | fSDE | 0.422 | 0.333 | 0.451 |
| **Stock** | Real | 0.000 | 0.000 | 12.655 |
| | JDFlow | 0.022 | 0.007 | 11.423 |
| | Fourier Flow | 0.144 | 0.010 | 12.569 |
| | PAR | 2.351 | 0.143 | 12.704 |
| | fSDE | 5.833 | 0.124 | 13.476 |

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_26

## 4 Conclusion

Our research has investigated the use of generative methods to model multivariate time series data. Our goal was to contribute valuable insights by evaluating generative models and introducing a neural SDE method driven by jump diffusion. We have proposed a novel normalising flow architecture that utilises a multivariate structure. Additionally, we have proposed a method for calibrating the parameters of the Merton jump diffusion model within our generative framework. We also proposed the new evaluation method. Code and training parameters for test models are available by the link https://github.com/kirillzx/JDFlow.

## References

1. Cohen, S.N., Reisinger, C., Wang, S.: Arbitrage-free neural-sde market models. Applied Mathematical Finance 30(1), 1–46 (2023)
2. Esteban, C., Hyland, S.L., Rätsch, G.: Real-valued (medical) time series generation with recurrent conditional gans. arXiv preprint arXiv:1706.02633 (2017)
3. Tang, F.: Merton jump-diffusion modeling of stock price data (2018)
4. Kobyzev, I., Prince, S.J., Brubaker, M.A.: Normalizing flows: An introduction and review of current methods. IEEE transactions on pattern analysis and machine intelligence 43(11), 3964–3979 (2020)
5. Geiger, A., Liu, D., Alnegheimish, S., Cuesta-Infante, A., Veeramachaneni, K.: Tadgan: Time series anomaly detection using generative adversarial networks. In: 2020 IEEE International Conference on Big Data (Big Data). pp. 33–43. IEEE (2020)
6. Wen, Q., Sun, L., Yang, F., Song, X., Gao, J., Wang, X., Xu, H.: Time series data augmentation for deep learning: A survey. arXiv preprint arXiv:2002.12478 (2020)
7. Øksendal, B., Øksendal, B.: Stochastic differential equations. Springer (2003)
8. Liu, X., Xiao, T., Si, S., Cao, Q., Kumar, S., Hsieh, C.J.: Neural sde: Stabilizing neural ode networks with stochastic noise. arXiv preprint arXiv:1906.02355 (2019)
9. Liu, W., Mao, X.: Strong convergence of the stopped euler–maruyama method for nonlinear stochastic differential equations. Applied Mathematics and Computation 223, 389–400 (2013)
10. Domingo, D., d'Onofrio, A., Flandoli, F.: Properties of bounded stochastic processes employed in biophysics. Stochastic Analysis and Applications 38(2), 277–306 (2020)
11. Alaa, A., Chan, A.J., van der Schaar, M.: Generative time-series modeling with fourier flows. In: International Conference on Learning Representations (2020)
12. Hayashi, K., Nakagawa, K.: Fractional sde-net: Generation of time series data with long-term memory. In: 2022 IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA). pp. 1–10. IEEE (2022)
13. Zhang, K., Patki, N., Veeramachaneni, K.: Sequential models in the synthetic data vault (2022)
14. Zakharov, K., Stavinova, E., Boukhanovsky, A.: Synthetic financial time series generation with regime clustering. Journal of Advances in Information Technology 14(6) (2023)
15. Yoon, J., Jarrett, D., Van der Schaar, M.: Time-series generative adversarial networks. Advances in neural information processing systems 32 (2019)
16. Morrill, J., Fermanian, A., Kidger, P., Lyons, T.: A generalised signature method for multivariate time series feature extraction. arXiv preprint arXiv:2006.00873 (2020)

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63759-9_26