# AIML425 ASSIGNMENT 5

*Benjamin McEvoy*

300579954

## 1. INTRODUCTION

This document revises the concepts of Diffusion, including models of Stochastic Differential Equation (SDE), Ordinary Differential Equation (ODE), the Euler-Murayama approach, and the Interpolation approach. The code is located in this repository or this Google Drive link: https://colab.research.google.com/drive/1OW2bRyrZbz_mX48lNhQsD8-zvovLyot1?usp=sharing.

## 2. THEORY

This section outlines the core knowledge of the assignment, and course for this program.

### 2.1. Diffusion

Diffusion in the context of neural networks, refers to the process that gradually transforms a simple distribution (i.e. Gaussian noise) into a more complex distribution, such as the distribution of real data. The concept of diffusion is utilised in tasks such as image generation and de-noising.

#### *2.1.1. Ordinary Differential Equation (ODE)*

An ordinary differential equation (ODE) is a differential equation that involves functions of a single variable and its derivatives. ODEs are used to model a wide range of fields such as physics, engineering, biology, and economics, where the behaviour of a system can be described in terms of rates of change.

An ODE is an equation that relates a function to its derivatives. The function typically represents a physical quantity, while the derivatives represent rates of change of that quantity. They can be described as:

$$dx = \left( f(x,t) - \frac{1}{2} g^2(t) \nabla_x \log(p(x,t)) \right) dt \tag{1}$$

$$\frac{dx}{dt} = f(x,t) - \frac{1}{2} g^2(t) \nabla_x \log(p(x,t)). \tag{2}$$

#### *2.1.2. Interpolation Approach*

The Interpolation Approach estimates unknown values between two or more known data points. Interpolation is utilised to predict or estimate values at points that lie within the range of a discrete set of known values. It allows for the creation of a continuous function or dataset from a limited number of observations. The simplest form is where the interpolated value is estimated using a straight line connecting two adjacent data points. For two points $x_0, y_0$ and $x_1, y_1$ the formula is:

$$y = y_0 + \frac{(y_1 - y_0)}{(x_1 - x_0)}(x - x_0)$$

#### *2.1.3. Stochastic Differential Equation*

A stochastic differential equation (SDE) is a type of differential equation that describes the behaviour of systems influenced by random noise or uncertainty. Compared to ODEs, which model deterministic processes, SDEs incorporate randomness. They combine deterministic and stochastic components to capture the inherent uncertainty in various real-world processes.

The SDE can be explain in mathematical form as:

$$dX_t = a(X_t, t)dt + b(X_t, t)dW_t$$

As SDEs do not have closed-form solutions, the Eular-Maruyama Method is used.

#### *2.1.4. Euler-Murayama Approach*

The Euler-Murayama approach is a numerical method to solve SDEs. It is an extension of the classical Euler method for ODEs, adapted to handle the randomness inherent in SDEs, and is particularly useful in simulating stochastic processes. It incorporates stochastic integrals, which are necessary for dealing with the random components of SDEs, and discretises time in the context of SDEs. This involves stepping through time in small increments and updating the state of the system based on both deterministic and stochastic terms. However, it may not provide high accuracy for all types of SDEs, particularly those with strong nonlinearities or significant volatility.

The formula can be described as:

$$X_{t+\Delta t} = X_t + f(X_t, t)\Delta t + g(X_t, t)\Delta W_t$$

## 3. EXPERIMENTS

This outlines the methodical approach to obtain the results, and explanations and conclusions about said results.

### 3.1. Methods

The explanation of the methods used in the code base, and results they provided.

#### *3.1.1. Euler-Murayama Approach*

The function implements the Euler method for updating the state of a system based on the score. It takes the current state, the score model that estimates the score of the data, the current time, the time increment, and optional scaling factors for noise and score to test the hyperparameters. The score is computed using the model, and noise is added to the system with a specified variance. The function then performs an Euler update step, which combines the current state, the scaled score, and the noise to produce the next state.

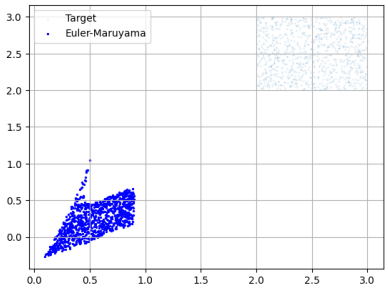
#### *3.1.2. Interpolation Approach*

The function is designed to perform a single step of the ODE update using a neural network to compute the velocity. The neural network is used to compute the velocity based on the current state and time, and the velocity is detached from the computation graph and converted back to an array. Finally, the function returns the updated state by adding the product of the velocity and the time increment $dt$ to the current state x. This effectively advances the state $x$ by one time step according to the estimated velocity.

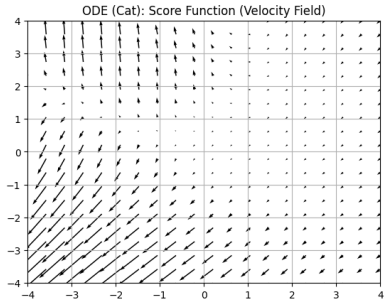
### 3.2. Results

The ODE approach maintains the shape of the target much better as seen by Figure 1. However, the steps taken push it towards a left downwards diagonal way and are unsuccessful in replicating the correct distribution in the right coordinates.

The Mean Squared Error (MSE) and Structural Similarity Index Measure (SSIM) metrics were used to compare the ODE and SDE approaches. The ODE method produced an MSE of 7.37 and an SSIM of 0.0413, while the SDE method had an MSE of 10.1 and a higher SSIM of 0.0871. These results suggest that while the ODE method had lower training losses, the SDE method exhibited better structural similarity in output quality. The SDE's higher SSIM indicates that it may be capturing more of the relevant features of the data, which could be beneficial in applications requiring perceptual quality, despite its higher MSE. In the comparison between the SDE and ODE functions in the graph output, the ODE maintains a closer distribution to the dog target rather than the SDE, as seen in Figure 2 (in Appendix).




(a) Shape of Output




(b) Velocity Chart

**Fig. 1**: Dog Distribution to Cat Distribution

## 4. TOOLS

This assignment used the tools of Numpy [1], Matplotlib [2], Seaborn [3], SKLearn [4] and Pytorch [5] to carry out experimentation.

Google Colab [6] is where the code base is hosted and is provided with any available machine during runtime allocation.

Besides the utilisation of these tools [7], I declare that the code and paper provided is my work.

## 5. CONCLUSION

Overall, the findings produced imply that while both ODE and SDE approaches can effectively reduce training loss, the choice between ODE and SDE methods may depend on the application's specific requirements, such as accuracy versus perceptual fidelity.

The code provided would need much-needed work to provide a more successful representation of the core concepts discussed and used, however for the context of the briefed assignment it shows the ODE method offering better numerical accuracy in reproducing the target distribution, making it potentially advantageous in certain applications. However, it highlights the importance of context and requirements when selecting a method for diffusion processes in neural networks.

## 6. REFERENCES


[1] Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al., "Array programming with numpy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[2] John D Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[3] Michael L. Waskom, "Seaborn: statistical data visualization," https://seaborn.pydata.org, 2021, Accessed: 2024-09-13.

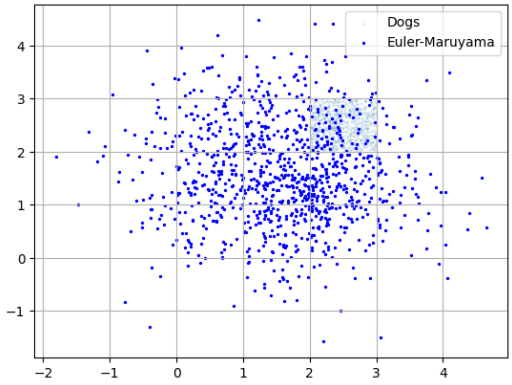
[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[5] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al., "Pytorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

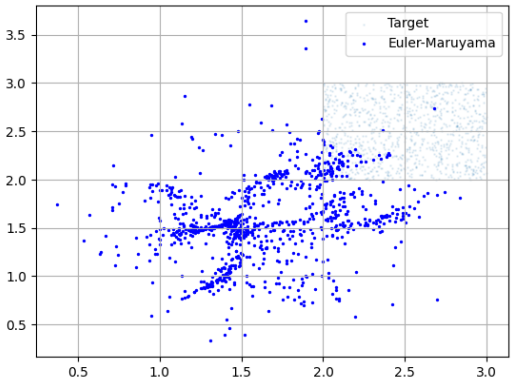
[6] Google, "Google colaboratory," Online, 2023, Available at: https://colab.research.google.com.

[7] Inc. Grammarly, "Grammarly," https://www.grammarly.com, 2024, Accessed: 2024-08-12.

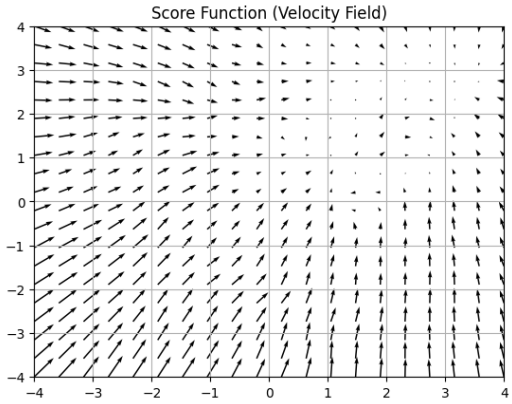

# Appendix



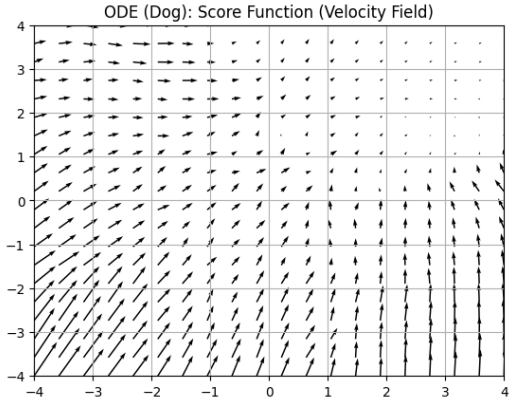

(a) Shape of SDE




(b) Shape of ODE




(c) Velocity Chart of SDE




(d) Velocity Chart of ODE

**Fig. 2**: Comparison between SDE and ODE Approaches