NEURAL CONTROLLED DIFFERENTIAL EQUATIONS WITH QUANTUM HIDDEN EVOLUTIONS

Lingyi Yang & Zhen Shao
Mathematical Institute
University of Oxford
{yangl, shaoz}@maths.ox.ac.uk

ABSTRACT

We introduce a class of neural controlled differential equation inspired by quantum mechanics. Neural quantum controlled differential equations (NQDEs) model the dynamics by analogue of the Schrödinger equation. Specifically, the hidden state represents the wave function, and its collapse leads to an interpretation of the classification probability. We implement and compare the results of four variants of NQDEs on a toy spiral classification problem.

1 INTRODUCTION

Controlled differential equations (CDEs) model the dynamics of a sequential output $Y_t \in \mathbb{R}^p$ in response to a sequential input $X_t \in \mathbb{R}^q$ by

$$dY_t = f(Y_t, t)dX_t, \tag{1}$$

for some fixed vector field $f(Y_t, t)$. Neural controlled differential equations (NCDEs) introduced by Kidger et al. (2021a) learn a CDE for a hidden variable z_t. The vector field $f(z_t, t)$ is modelled by a neural network $f(z_t, t; \theta)$ where θ are learnable parameters. The output is then $Y_t = l(z_t)$, for some linear function l. The parameters θ are fitted by the given input sequences X_t and the outputs Y_t.

Probabilistic models are popular for alleviating the issue of overfitting, motivating heuristics such as dropout. Quantum mechanics have probabilistic interpretation and is key in modelling physical phenomena. We introduce architectures inspired by the Schrödinger equation to model the latent space of NCDEs. Analogous to the Born interpretation for the collapse of the wave function, we have a *collapse* function for observation times (where we need to make an inference for Y_t). As quantum systems and unitary systems (where the transition is unitary i.e. $U^*U = I$) are intricately linked, we implement and apply four variants to a toy spiral classification problem.

2 NEURAL QUANTUM DIFFERENTIAL EQUATIONS

Complex numbers are indispensable for quantum mechanics. Complex recurrent neural networks, e.g. uRNN (Arjovsky et al., 2016) and ceRNN (Shafran et al., 2019), have been studied to derive stability and convergence results. Barrachina et al. (2023) provides a Python library for complex neural networks complete with backpropogation and other real neural network equivalents like max-pooling for complex functions. For other related literature, see Appendix C.

Combining neural controlled differential equations with quantum concepts has not yet been explored. We introduce a family of models based on quantum mechanical postulates. In particular, in the quantum world, the state of a system is represented by a complex wave function $\psi(x, t)$. Measurements/observables are modelled as linear operators on $\psi(x, t)$. The set of possible outcomes of these measurements are eigenvalues of this operator. The probability of obtaining any particular eigenvalue as the observation is proportional to the inner product between its associated eigenvector and the state $\psi(x, t)$. Therefore, the normalised inner products represent a probability distribution over the states. The hidden quantum state evolves according to the Schrödinger equation (Dirac, 1981)

$$d\psi(x, t) = -i\hbar H\psi(x, t)dt, \tag{2}$$

where $\hbar$ is the normalised Planck constant and H is the Hamiltonian. The evolution of the quantum state is a unitary operator, and the exponential of a time-independent Hamiltonian generates this unitary operator.

Our model is inspired by the practical success of neural differential equation based models, and the success of the quantum physics postulate to model physical systems. To obtain a neural quantum controlled differential equation (NQDE), we suppose that the dynamics of the latent state z_t is modelled by the Schrödinger equation, driven by some control path X_t, that is,

$$dz_t = -i\hbar H z_t dX_t. \tag{3}$$

Note that $-i\hbar H z_t$ is playing the role of the vector field f in the CDE (1). For this complex vector field, $-i\hbar H z_t$, unitary conditions are imposed. Unitary matrices are known to have good stability properties. ExpRNNs (Lezcano-Casado & Martínez-Rubio, 2019) ensure orthogonality/unitarity using the exponential map from Lie group theory. The projUNN method developed by Kiani et al. (2022) projects the updated matrix back into the class of unitary matrices.

For each time that we need to "observe" the hidden state and make an inference for the output Y_t, the hidden states pass through an operation that we refer to as the *collapse*. For a classification problem with m classes, the collapse function is given by $g : \mathbb{C}^m \to \mathbb{R}^m$ or equivalently $\tilde{g} : \mathbb{R}^{2m} \to \mathbb{R}^m$, where $\tilde{g}$ is composed of $g_1 : \mathbb{R}^{2m} \to \mathbb{R}^m$, $g_2 : \mathbb{R}^m \to \mathbb{R}^m$, and $g_3 : \mathbb{R}^m \to \mathbb{R}^m$. The function g_1 takes the squared modulus of the complex input (represented in the real space). Then g_2 normalises the output from g_1 to have norm 1, thus the output of g_2 is a probability distribution, which we can then sample (g_3) for the output Y_t. Therefore

$$Y_t = \tilde{g}(z_t) = g_3(g_2(g_1(z_t))).$$

This is analogous to the quantum system collapsing to an eigenstate. We see that in practice a softmax can be utilised for g_2.

3 Experimental results

We look a classification problem on the bi-directional spiral dataset as detailed by Chen et al. (2018) using 128 spirals. We implemented four vector fields. Two of these have unitary constraints as imposed by ProjUNN (Kiani et al., 2022) (and denoted by _unn in the name) and the other two with orthogonal constraints using GeoTorch (Lezcano-Casado, 2019) (denoted by _geo in the name). For each constraint method, we look at two variants: the first looks at modelling each class of the classification task separately, then concatenates the results together before a final linear layer (NQDE1_unn and NQDE3_geo) and the second performs the concatenation after the linear layer (NQDE2_unn and NQDE4_geo). For specific architectures that we utilised and the number of trainable parameters, see Appendix A. The results are given in Table 1. Hyperparameters can be found in Appendix B.

model	**final loss**	**forward NFE**	**backward NFE**	**accuracy**
NQDE1_unn	**0.00028 (0.0004)**	**1069.79 (58.71)**	**2337.67 (431.67)**	1.000 (0)
NQDE2_unn	0.00717 (0.0111)	1102.86 (54.95)	3093.38 (489.91)	1.000 (0)
NQDE3_geo	0.12472 (0.2095)	1348.19 (43.75)	6781.65 (1690.76)	1.000 (0)
NQDE4_geo	0.03786 (0.0167)	1288.21 (325.52)	4425.68 (1561.47)	1.000 (0)

Table 1: Spiral classification results on various architectures. Standard deviation in brackets are reported over 3 repeats.

The models all use 20 epochs so that we can compare data efficiency. Given very limited data of only 128 spirals, we see that all of these architectures learn relevant dynamics for spiral classification and can reach 100% accuracy after hyperparameter optimisation. Using orthogonal linear layers with GeoTorch requires more function evaluations. This is not unexpected as ProjUNN architectures makes a rank-k approximation for computational efficiency. Using ProjUNN with the concatenation occurring before the linear layer gives the best model in terms of both loss and has the smallest number of function evaluations (NFEs). The code for the experiments can be found at the Github repository https://github.com/lingyiyang/NQDE.

4 Conclusion/Discussion

We have demonstrated that neural controlled differential equation architectures that emulate quantum evolutions can learn relevant dynamics on a toy classification problem. For future work, we would like to explore the approximation power of these models in greater depths (to derive similar results to Voigtlaender (2023)) as well as compare with other models on larger datasets.

ACKNOWLEDGEMENTS

L.Y. was supported by the Alan Turing Institute and by the EPSRC [EP/S026347/1]. Z. S. was supported by the EPSRC [EP/S026347/1].

URM STATEMENT

The authors acknowledge that at least one key author of this work meets the URM criteria of ICLR 2024 Tiny Papers Track.

REFERENCES

Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1120–1128, New York, New York, USA, 20–22 Jun 2016. PMLR.

Jose Agustin Barrachina, Chengfang Ren, Gilles Vieillard, Christele Morisseau, and Jean-Philippe Ovarlez. Theory and implementation of complex-valued neural networks. *arXiv preprint arXiv:2302.08286*, 2023.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. Gru-ode-bayes: Continuous modeling of sporadically-observed time series. *Advances in neural information processing systems*, 32, 2019.

Paul Adrien Maurice Dirac. *The principles of quantum mechanics*. Number 27. Oxford university press, 1981.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021a.

Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state space layers. *Advances in neural information processing systems*, 34:572–585, 2021b.

Junteng Jia and Austin R Benson. Neural jump stochastic differential equations. *Advances in Neural Information Processing Systems*, 32, 2019.

Bobak Kiani, Randall Balestriero, Yann LeCun, and Seth Lloyd. projunn: efficient method for training deep networks with unitary matrices. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 14448–14463. Curran Associates, Inc., 2022.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6696–6707. Curran Associates, Inc., 2020.

Patrick Kidger, James Foster, Xuechen Li, and Terry J Lyons. Neural sdes as infinite-dimensional gans. In *International conference on machine learning*, pp. 5453–5463. PMLR, 2021a.

Patrick Kidger, James Foster, Xuechen Chen Li, and Terry Lyons. Efficient and accurate gradients for neural sdes. *Advances in Neural Information Processing Systems*, 34:18747–18761, 2021b.

ChiYan Lee, Hideyuki Hasegawa, and Shangce Gao. Complex-valued neural networks: A comprehensive survey. *IEEE/CAA Journal of Automatica Sinica*, 9(8):1406–1426, 2022.

Mario Lezcano-Casado. Trivializations for gradient-based optimization on manifolds. In *Advances in Neural Information Processing Systems, NeurIPS*, pp. 9154–9164, 2019.

Mario Lezcano-Casado and David Martínez-Rubio. Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group. In *International Conference on Machine Learning (ICML)*, pp. 3794–3803, 2019.

Qiuchi Li, Dimitris Gkoumas, Alessandro Sordoni, Jian-Yun Nie, and Massimo Melucci. Quantum-inspired neural network for conversational emotion recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 13270–13278, 2021.

Xuanqing Liu, Tesi Xiao, Si Si, Qin Cao, Sanjiv Kumar, and Cho-Jui Hsieh. Neural sde: Stabilizing neural ode networks with stochastic noise. *arXiv preprint arXiv:1906.02355*, 2019.

James Morrill, Patrick Kidger, Lingyi Yang, and Terry Lyons. Neural controlled differential equations for online prediction tasks. *arXiv preprint arXiv:2106.11028*, 2021a.

James Morrill, Cristopher Salvi, Patrick Kidger, and James Foster. Neural rough differential equations for long time series. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 7829–7838. PMLR, 18–24 Jul 2021b.

Alexander Norcliffe, Cristian Bodnar, Ben Day, Jacob Moss, and Pietro Liò. Neural ode processes. *arXiv preprint arXiv:2103.12413*, 2021.

Viktor Oganesyan, Alexandra Volokhova, and Dmitry Vetrov. Stochasticity in neural odes: An empirical study. *arXiv preprint arXiv:2002.09779*, 2020.

Sourav Pal, Zhanpeng Zeng, Sathya N Ravi, and Vikas Singh. Controlled differential equations on long sequences via non-standard wavelets. 2023.

Yulia Rubanova, Ricky T. Q. Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

T Konstantin Rusch and Siddhartha Mishra. Unicornn: A recurrent model for learning very long time dependencies. In *International Conference on Machine Learning*, pp. 9168–9178. PMLR, 2021.

Cristopher Salvi, Maud Lemercier, and Andris Gerasimovics. Neural stochastic pdes: Resolution-invariant learning of continuous spatiotemporal dynamics. *Advances in Neural Information Processing Systems*, 35:1333–1344, 2022.

Mona Schirmer, Mazin Eltayeb, Stefan Lessmann, and Maja Rudolph. Modeling irregular time series with continuous recurrent units. In *International Conference on Machine Learning*, pp. 19388–19405. PMLR, 2022.

Izhak Shafran, Tom Bagby, and R. J. Skerry-Ryan. Complex evolution recurrent neural networks (cernns). 2019.

Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. *arXiv preprint arXiv:2208.04933*, 2022.

Felix Voigtlaender. The universal approximation theorem for complex-valued neural networks. *Applied and Computational Harmonic Analysis*, 64:33–61, 2023. ISSN 1063-5203. doi: https://doi.org/10.1016/j.acha.2022.12.002.

Hedi Xia, Vai Suliafu, Hangjie Ji, Tan Nguyen, Andrea Bertozzi, Stanley Osher, and Bao Wang. Heavy ball neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 34:18646–18659, 2021.

A MODEL ARCHITECTURE

In this Appendix, we give details about the structure of the four NQDE models (for the vector field of the controlled differential equation).

Using ProjUNN, the model for the first architecture where we combine the representation of complex values (for each class) before the final linear layer is seen below. This model has 5052 trainable model parameters.

```
NeuralQDE(
  (func): QDEFunc(
    (linear1): Linear(in_features=4, out_features=32, bias=True)
    (linear2): Linear(in_features=64, out_features=12, bias=True)
    (rnn_layer): OrthogonalRNN(
      (recurrent_kernel): Linear(in_features=32, out_features=32,
        ↪ bias=False)
      (input_kernel): Linear(in_features=32, out_features=32,
        ↪ bias=False)
      (nonlinearity): ReLU()
    )
    (rnn_layer2): OrthogonalRNN(
      (recurrent_kernel): Linear(in_features=32, out_features=32,
        ↪ bias=False)
      (input_kernel): Linear(in_features=32, out_features=32,
        ↪ bias=False)
      (nonlinearity): ReLU()
    )
  )
  (initial): Linear(in_features=3, out_features=4, bias=True)
)
```

Using ProjUNN, the model for the second architecture where we combine the representation of complex values (for each class) after the final linear layer is seen below. This model has 4470 trainable model parameters.

```
NeuralQDE(
  (func): QDEFunc2(
    (linear1): Linear(in_features=4, out_features=32, bias=True)
    (linear2): Linear(in_features=32, out_features=6, bias=True)
    (rnn_layer): OrthogonalRNN(
      (recurrent_kernel): Linear(in_features=32, out_features=32,
        ↪ bias=False)
      (input_kernel): Linear(in_features=32, out_features=32,
        ↪ bias=False)
      (nonlinearity): ReLU()
    )
    (rnn_layer2): OrthogonalRNN(
      (recurrent_kernel): Linear(in_features=32, out_features=32,
        ↪ bias=False)
      (input_kernel): Linear(in_features=32, out_features=32,
        ↪ bias=False)
      (nonlinearity): ReLU()
    )
  )
  (initial): Linear(in_features=3, out_features=4, bias=True)
)
)
```

Using GeoTorch, the model for the third architecture where we combine the representation of complex values (for each class) before the final linear layer is seen below. This model has 3068 trainable model parameters.

```
NeuralQDE(
  (func): QDEFunc3(
    (linear1): Linear(in_features=4, out_features=32, bias=True)
    (linear2): ParametrizedLinear(
      in_features=32, out_features=32, bias=True
      (parametrizations): ModuleDict(
        (weight): ParametrizationList(
          (0): Stiefel(n=32, k=32, triv=linalg_matrix_exp)
        )
      )
    )
    (linear3): ParametrizedLinear(
      in_features=32, out_features=32, bias=True
      (parametrizations): ModuleDict(
        (weight): ParametrizationList(
          (0): Stiefel(n=32, k=32, triv=linalg_matrix_exp)
        )
      )
    )
    (linear4): Linear(in_features=64, out_features=12, bias=True)
  )
  (initial): Linear(in_features=3, out_features=4, bias=True)
)
```

Using GeoTorch, the model for the fourth architecture where we combine the representation of complex values (for each class) after the final linear layer is seen below. This model has 2486 trainable model parameters.

```
NeuralQDE(
  (func): QDEFunc4(
    (linear1): Linear(in_features=4, out_features=32, bias=True)
    (linear2): ParametrizedLinear(
      in_features=32, out_features=32, bias=True
      (parametrizations): ModuleDict(
        (weight): ParametrizationList(
          (0): Stiefel(n=32, k=32, triv=linalg_matrix_exp)
        )
      )
    )
    (linear3): ParametrizedLinear(
      in_features=32, out_features=32, bias=True
      (parametrizations): ModuleDict(
        (weight): ParametrizationList(
          (0): Stiefel(n=32, k=32, triv=linalg_matrix_exp)
        )
      )
    )
    (linear4): Linear(in_features=32, out_features=6, bias=True)
  )
  (initial): Linear(in_features=3, out_features=4, bias=True)
)
```

B Hyper-parameter settings

The hyperparamter of the experiments can be seen in Table 2.

C Related neural differential equation work

Seen as a continuous time generalisation of discrete models, differential equation based neural networks enjoyed successes in recent years. Chen et al. (2018) proposed neural ordinary differential equations (ODEs) and demonstrated its performance on classification and time-series generation

model	epoch	lr	lin_size
NQDE1_unn	20	0.002	32
NQDE2_unn	20	0.002	32
NQDE3_geo	20	0.001	32
NQDE4_geo	20	0.001	32

Table 2: Hyperparameter settings for the four models

problems. In a follow-up work Rubanova et al. (2019) uses a neural ODE to model the (autonomous) evolution of the hidden state before the next observation arrives, and obtained encouraging results. This approach is then generalised to the neural controlled differential equations (CDEs) (Kidger et al., 2020). Kidger et al. (2021a) also worked on an extension to neural stochastic differential equations (SDEs) and application to GANs for learning path distribution. Neural rough differential equations (RDEs) are proposed as an improvement on neural CDEs by summarising sub-intervals using log-signatures, deriving the theoretical justifications from the log-ODE method (Morrill et al., 2021b). Pal et al. (2023) proposed another way of adapting neural CDEs to handle long time series.

Heavy ball method is proposed to improve training of neural ODE in (Xia et al., 2021). Norcliffe et al. (2021) combines neural ODEs with the so-called neural processes so that the model (the trained neural network) can adapted to incoming data stream. Morrill et al. (2021a) uses neural CDEs for online prediction task. Jia & Benson (2019) extends neural ODEs to handle stochastic jumps and discussed training techniques when the latent state has discontinuities. Oganesyan et al. (2020) and Liu et al. (2019) view neural SDEs as a stochastic regularisation technique for training neural ODEs and evaluated empirically the performances. Kidger et al. (2021b) proposed improved technique for training of neural SDEs. Salvi et al. (2022) proposed using neural network to learn the dynamics of stochastic partial differential equations (SPDEs) and showed that for several well-known SPDE dynamics, the solver can be learned faster than traditional numerical solvers of SPDE. Deep state space models such as the S4 (Gu et al., 2021a;b) can effectively model long range dependency in sequence modelling, this is subsequently improved by Smith et al. (2022).

The survey (Lee et al., 2022) discuses split-complex neural networks, which split the complex value input into real and imaginary parts which are fed into a real-valued neural network, that could have real-valued weight and real activation or complex-valued weights and real activation. Some training instability issues are highlighted. There are few applications of complex RNNs. De Brouwer et al. (2019) improves the variational autoencoder application of neural ODE by combining neural ODE with a continuous version of a GRU. Schirmer et al. (2022) uses an SDE with a Kalman filter to connect observations at different timestamps in an RNN. Rusch & Mishra (2021) studied a restricted class of ODE discretised RNN based on Hamiltonian system ODE. Li et al. (2021) implicitly made a connection between unitary RNNs and quantum-inspired theory, and experimented a unitary RNN on an emotional recognition problem from multi-modal time-series data.

In terms of function approximation power, Voigtlaender (2023) finds that unlike the classical case of real networks, the set of "good activation functions"—which give rise to networks with the universal approximation property—differs significantly depending on whether one considers deep networks or shallow networks. For deep networks with at least two hidden layers, the universal approximation property holds as long as σ, the activation function, is neither a polynomial, a holomorphic function, nor an antiholomorphic function. Shallow networks, on the other hand, are universal if and only if the real part or the imaginary part of σ is not a polyharmonic function.