A Study on Adaptive Penalty Functions in Neural ODEs for Real Systems Modeling

C. Coelho[1,a)], M. Fernanda P. Costa[1,b)] and L.L. Ferrás[1,2,c)]

[1]*Centre of Mathematics (CMAT), University of Minho, Braga, 4710-57, Portugal.*
[2]*Department of Mechanical Engineering (Section of Mathematics) - FEUP, University of Porto, Porto, 4200-465, Portugal*

[a)]Corresponding author: cmartins@cmat.uminho.pt
[b)]mfc@math.uminho.pt
[c)]lferras@fe.up.pt

Abstract. Neural Ordinary Differential Equations (Neural ODEs) have emerged as a powerful tool for modeling the dynamics of systems using continuous-time functions. However, the current optimization framework for Neural ODEs is based on unconstrained optimization, which limits their ability to accurately model constrained real-world systems. In this work, we address this limitation by evaluating and analyzing three adaptive penalty functions with different parameter update strategies, providing insights into the impact of different parameter update strategies on the performance of Neural ODEs.

The code to replicate the experiments presented in this paper can be found at `https://github.com/CeciliaCoelho/PriorKnowledgeNeuralODE`.

INTRODUCTION

Mathematical modeling of real-world systems has significant advantages in terms of prediction, understanding and simulation. In recent times, due to the impressive results demonstrated by neural networks (NNs), Neural ODEs have emerged as a promising approach to model continuous-time systems [1].

In many instances, real-world systems follow various laws or constraints that must be considered when fitting a mathematical model. Failing to account for such constraints may result in unrealistic mathematical models and consequently implausible predictions, that deviate significantly from the observed reality. This discrepancy can undermine the credibility of the model and hinder its practical applicability. Furthermore, by incorporating prior knowledge constraints, a model is better equipped to capture the underlying laws and behavior of a system instead of purely relying on quantity and quality of the available training data. Traditionally, NNs and Neural ODEs formulate the problem of modeling a given system as an unconstrained optimization problem by minimizing the distance between the ground-truth data and the model predictions. To incorporate the constraints that govern a real system into the modeling process, it is necessary to formulate it as a constrained optimization problem.

A real-world system is characterized by a time-series comprising observations at discrete time steps t_i ($i = 1, \ldots, N$) and formulated as a constrained problem as follows:

$$\begin{aligned}
\underset{\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}}{\text{minimize}} \quad & \langle l_t(\boldsymbol{\theta}) \rangle = \frac{1}{N} \sum_{t=t_1}^{t_N} l_t(\boldsymbol{\theta}) \\
\text{subject to} \quad & \boldsymbol{c}_t^j(\boldsymbol{\theta}) = 0, \quad j \in \varepsilon, \quad t = t_1, \ldots, t_N, \\
& \boldsymbol{c}_t^i(\boldsymbol{\theta}) \leq 0, \quad i \in \mathcal{I}, \quad t = t_1, \ldots, t_N,
\end{aligned} \tag{1}$$

where $\langle l_t(\boldsymbol{\theta}) \rangle : \mathbb{R}^{n_\theta} \to \mathbb{R}$ is the objective function given by the average of the difference between the ground-truth and the predicted values ($l_t(\boldsymbol{\theta})$) at all time steps (with n_θ the number of parameters $\boldsymbol{\theta}$ of the NN), $\boldsymbol{c}_t^j, \boldsymbol{c}_t^i : \mathbb{R}^{n_\theta} \to \mathbb{R}$ are the equality and inequality constraint functions with ε the equality and $\mathcal{I}$ inequality index sets of constraints, respectively.

The set of points $\boldsymbol{\theta}$ that satisfy all the constraints defines the feasible set $\mathcal{S} = \{\boldsymbol{\theta} \in \mathbb{R}^{n_\theta} : \boldsymbol{c}_t^j(\boldsymbol{\theta}) = 0,\ j \in \varepsilon;\ \boldsymbol{c}_t^i(\boldsymbol{\theta}) \leq 0,\ i \in \mathcal{I}, t = t_1, \ldots, t_N\}$.

Penalty methods have emerged as the most common technique for handling constrained problems due to their simplicity and ease of implementation.

In the context of penalty methods, the constrained optimization problem (1) is reformulated as an unconstrained problem by incorporating penalty terms into the objective function. The penalty function $\phi(\boldsymbol{\theta})$ combines the objective function and measures of constraints violation as follows:

$$\underset{\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}}{\text{minimize}} \quad \phi(\boldsymbol{\theta}) = \langle l_t(\boldsymbol{\theta}) \rangle + \frac{\mu_j}{2} P_j(\boldsymbol{\theta}) + \frac{\mu_i}{2} P_i(\boldsymbol{\theta})$$

where $\mu_j, \mu_i > 0$ are the penalty parameters for the penalty terms P_j, P_i that measure the equality and inequality constraints violations, respectively.

By making μ_j and μ_i to infinity, the constraints violation are penalized with increased severity. The penalty parameters play a crucial role as they determine the trade-off between the objective function and the constraints violation. Larger values of μ_j, μ_i emphasize the importance of satisfying the constraints, which can result in more accurate solutions. Inversely, smaller values of μ_j, μ_i may prioritize the minimization of the objective function, potentially leading to less constraint satisfaction. Choosing the initial values of μ_j, μ_i and updating its values through the iterative process is a challenging task that requires careful consideration.

To address this challenge, adaptive penalty methods that aim to automatically update the penalty parameters based on the satisfaction of constraints during the optimization process are proposed in the literature [2, 3, 4]. In this paper, we investigate and compare the performance of three different adaptive penalty functions for modeling two real-world constrained systems, namely the world population growth [5] and the evolution of a chemical reaction [6] using Neural ODEs.

ADAPTIVE PENALTY FUNCTIONS

We analyze and compare the performance of three adaptive penalty functions that have been proposed in the literature. In this work we used the penalty functions described in [2] and [3], herein denoted by *Adaptive 1* and *Adaptive 3*, respectively. In these penalty functions, the equality constraints $c_t^j = 0$ are considered as a pair of inequality constraints $c_t^j - \beta \leq 0$ and $-c_t^j - \beta \leq 0$, where $\beta > 0$ is a small tolerance. These penalty functions were developed for population-based algorithms, and therefore, some adaptations were necessary to use them in the training process of NNs for this study. The other penalty function was specifically defined for Neural ODEs and was proposed in [4], herein denoted by *Adaptive 2*.

***Adaptive 1*:** We adapted the function proposed in [2] to be used with NNs and to model real-world systems given by time-series data. To adapt this function, the terms l_t and the inequality constraints violations $\boldsymbol{v}_t^i$ are averaged for all time steps instead of averaged in the current population of points, and defined as:

$$\phi(\boldsymbol{\theta}) = \begin{cases} \langle l_t(\boldsymbol{\theta}) \rangle, & \text{if } \boldsymbol{\theta} \in \mathcal{S}, \\ \langle l_t(\boldsymbol{\theta}) \rangle + \sum_{i \in \mathcal{I}} \mu_i \langle \boldsymbol{v}_t^i(\boldsymbol{\theta}) \rangle, & \text{if } \boldsymbol{\theta} \notin \mathcal{S}, \end{cases}$$

where $\langle z \rangle$ denotes the average of set z. The penalty parameters μ_i are computed as a measure of the ratio between the average constraint violation i at all time steps and the sum of the squares of the averages of all constraint violations at all time steps, at the current iteration. These penalty parameters are further multiplied by the objective function:

$$\mu_i = \langle l_t(\boldsymbol{\theta}) \rangle \frac{\langle \boldsymbol{v}_t^i(\boldsymbol{\theta}) \rangle}{\sum_{i \in \mathcal{I}} \left(\langle \boldsymbol{v}_t^i(\boldsymbol{\theta}) \rangle \right)^2}, \ i \in \mathcal{I}$$

***Adaptive 2*:** The penalty function was proposed in [4] to enable modeling constrained systems with Neural ODEs. In this penalty function, the objective function and the constraints violations are normalized using a bounded function $\psi(x) = 1 - \dfrac{1}{1+x}$ to ensure that all values have the same order of magnitude, and is defined by:

$$\phi(\boldsymbol{\theta}) = \begin{cases} \langle l_t(\boldsymbol{\theta})\rangle, & \text{if } \boldsymbol{\theta} \in \mathcal{S}, \\ \langle l_t(\boldsymbol{\theta})\rangle + \dfrac{1}{\#\{\mathcal{E}\}} \sum_{j\in\mathcal{E}} \mu_j \langle \boldsymbol{v}_t^j(\boldsymbol{\theta})\rangle + \dfrac{1}{\#\{\mathcal{I}\}} \sum_{i\in\mathcal{I}} \mu_i \langle \boldsymbol{v}_t^i(\boldsymbol{\theta})\rangle, & \text{if } \boldsymbol{\theta} \notin \mathcal{S}, \end{cases}$$

The penalty parameters of the equality μ_j and inequality μ_i constraints violations are defined as the proportion of predictions that violate the constraint at all time steps at the current iteration:

$$\mu_i = \frac{\#\{t : \boldsymbol{v}_t^i \neq 0\}}{N}, \quad \mu_j = \frac{\#\{t : \boldsymbol{v}_t^j \neq 0\}}{N},$$

where $\#\{z\}$ denotes the cardinality of set z.

***Adaptive 3*:** We adapted the function proposed in [3] for time-series modeling with NNs. Thus, the terms l_t and the inequality constraints violations $\boldsymbol{v}_t^i$ are averaged for all time steps instead of averaged in the current population of points:

$$\phi(\boldsymbol{\theta}) = \langle l_t(\boldsymbol{\theta})\rangle + \sum_{i\in\mathcal{I}} \mu_i \langle \boldsymbol{v}_t^i(\boldsymbol{\theta})\rangle$$

where the penalty parameters μ_i are computed dynamically by increasing its values as the number of iterations progresses (itr). Two different updates were proposed (herein denoted by version 1 and version 2):

$$\textbf{(version 1)}\ \mu_i = (C \times \text{itr})^a \ \text{ OR } \ \textbf{(version 2)}\ \mu_i = D \times \text{itr}\sqrt{\text{itr}}$$

where $C, D, a > 0$. In the numerical experiments presented we set $C = 0.5, D = 1, a = 2$.

NUMERICAL EXPERIMENTS

The study of the performance of the three adaptive penalty functions was numerically conducted by modeling two systems with Neural ODEs: the world population growth through time, $P(t)$, with an inequality constraint, per time step, defined by the carrying capacity $P(t) \leq 12$ (WPG dataset) [5], and the evolution of the mass of four chemical species in a reaction: $m_A(t), m_B(t), m_C(t)$ and $m_D(t)$, with an equality constraint, per time step, defined by the conservation of mass $m_A(t)+m_B(t)+m_C(t)+m_D(t) = constant$ (CR dataset) [6]. For each system, three different experiments were performed: reconstruction (same dataset for training/testing); extrapolation (larger time horizon in the testing set); completion (same time interval for training /testing but higher frequency sampling in the testing set). Training was performed using Mean Squared Error (MSE) as the objective function, Adam optimizer with learning rate $1e-5$ and 10000 iterations. For the WPG, the NN has 4 hidden layers with 50 neurons each and the following activation functions, respectively: linear; hyperbolic tangent (tanh); linear; Exponential Linear Unit (ELU). The input and output layers have 1 neuron. For the CR the NN has 6 hidden layers: linear with 50 neurons; tanh with 50 neurons; linear with 64 neurons; ELU with 64 neurons; linear with 50 neurons; tanh with 50 neurons. The input and output layers have 4 neurons. Performance was analyzed by conducting three independent runs for each experiment and computing the average MSE (MSE_{avg}), average constraints violation (V_{avg}) and the respective standard deviation (std_{avg}).

The results of the performance evaluation using the three adaptive penalty functions, including the two penalty parameter update versions of *Adaptive 3*, are shown in Table 1 and Table 2 for the WPG and CR systems, respectively.

TABLE 1. Testing results of Neural ODEs trained using the adaptive penalty functions when modeling the WPG system.

	Adaptive 1		*Adaptive 2*		*Adaptive 3*			
					version 1		version 2	
Experiment	$MSE_{avg} \pm std_{avg}$	$V_{avg} \pm std_{avg}$	$MSE_{avg} \pm std_{avg}$	$V_{avg} \pm std_{avg}$	$MSE_{avg} \pm std_{avg}$	$V_{avg} \pm std_{avg}$	$MSE_{avg} \pm std_{avg}$	$V_{avg} \pm std_{avg}$
Reconstruction	4.485e-2 ± 3.593e-2	2.639e-2 ± 1.419e-2	**1.300e-3 ± 4.308e-4**	**6.898e-4 ± 2.310e-4**	35.696 ± 24.921	0.0 ± 0.0	50.551 ± 45.793	0.0 ± 0.0
Extrapolation	1.538e-1 ± 1.673e-2	1.729e-1 ± 6.527e-3	**1.411e-3 ± 1.800e-3**	**1.000e-2 ± 3.900e-4**	46.640 ± 5.441	0.0 ± 0.0	59.756 ± 42.081	0.0 ± 0.0
Completion	1.384e-2 ± 0.013603	1.249e-2 ± 9.871e-3	**8.903e-4± 5.00e-5**	**4.400e-4 ± 5.210e-5**	66.972 ± 8.573	0.0 ± 0.0	12.294 ± 9.635	0.0 ± 0.0

From Table 1, *Adaptive 2* yielded models with the lowest MSE_{avg} values across all experiments. On the other hand, *Adaptive 3* versions 1 and 2 achieved the lowest V_{avg} ($V_{avg} = 0$). However, in both versions of *Adaptive 3* exhibited extremely high MSE_{avg} values, meaning that the solutions are feasible but non-optimal. Overall, *Adaptive 2* demonstrated the best performance when modeling the WPG system.

TABLE 2. Testing results of Neural ODEs trained using the adaptive penalty functions when modeling the CR system.

	Adaptive 1		*Adaptive 2*		*Adaptive 3*			
					version 1		version 2	
Experiment	$MSE_{avg} \pm std_{avg}$	$V_{avg} \pm std_{avg}$	$MSE_{avg} \pm std_{avg}$	$V_{avg} \pm std_{avg}$	$MSE_{avg} \pm std_{avg}$	$V_{avg} \pm std_{avg}$	$MSE_{avg} \pm std_{avg}$	$V_{avg} \pm std_{avg}$
Reconstruction	**4.300e-5 ± 2.112e-5**	**1.200e-5 ± 7.121e-6**	2.001e-4± 2.000e-4	2.221e-6 ± 1.400e-6	8.475e-2 ± 2.146e-2	2.000e-6 ± 1.010e-6	2.523e-2 ± 1.066e-2	7.000e-6 ± 7.001e-6
Extrapolation	2.472e-1 ± 3.337e-1	2.794e-1 ± 2.3e-1	**8.001e-5 ± 2.020e-5**	**2.000e-5± 2.000e-5**	2.761e-2 ± 2.239e-1	8.140e-4 ± 0.001144	1.129e-1 ± 1.402e-2	4.910e-4 ± 6.120e-4
Completion	**3.400e-5 ± 7.000e-6**	**6.000e-6 ± 1.000e-6**	1.000e-3 ± 1.400e-3	1.410e-4 ± 2.003e-4	3.548e-2 ± 1.289e-2	1.100e-5 ± 3.001e-6	8.606e-2 ± 5.659e-2	2.730e-4 ± 3.620e-4

From Table 2, the models yielded by *Adaptive 1* demonstrates the best performance in 2 out of 3 experiments, namely for the reconstruction and completion. Note that, *Adaptive 3* with version 1 achieves a slightly lower V_{avg} value in the reconstruction, but it has higher MSE_{avg} value, meaning that the solutions are feasible but non-optimal. For the extrapolation experiment, *Adaptive 2* exhibits the best performance with smaller MSE_{avg} and V_{avg} values.

CONCLUSION

In this paper we present three adaptive penalty functions that are analyzed and evaluated when used in the training of Neural ODEs for modeling constrained real-world systems.

To evaluate and analyze the performance of these functions, two case studies were conducted: modeling the WPG and CR systems. The average MSE and constraints violation values were computed for the testing sets. The numerical results show that, overall, *Adaptive 2* delivers the best performance models. Additionally, the results demonstrate that the penalty parameter computation strategies employed by *Adaptive 3* (version 1 and version 2) is not able to deliver models that make good predictions. This study shows the potential of these adaptive penalty functions for modeling real-world systems with Neural ODEs.

In future research, we will further investigate the performance of *Adaptive 2* when training Neural ODEs for modeling more complex types of real-world systems.

ACKNOWLEDGMENTS

The authors acknowledge the funding by Fundação para a Ciência e Tecnologia (FCT) (Portuguese Foundation for Science and Technology) through CMAT projects UIDB/00013/2020 and UIDP/00013/2020. C. Coelho would like to thank FCT for the funding through the scholarship with reference 2021.05201.BD and the computing facilities provided by Project Search-ON2: Revitalization of HPC infrastructure of UMinho" (NORTE-7-162-FEDER-00086), co-funded by the North Portugal Regional Operational Programme (ON.2 – O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF).

REFERENCES

[1] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, Advances in neural information processing systems **31** (2018).

[2] A. C. C. Lemonge and H. J. C. Barbosa, International Journal for Numerical Methods in Engineering **59**, 703–736February (2004).

[3] R. B. Francisco, M. F. P. Costa, and A. M. A. Rocha, "A firefly dynamic penalty approach for solving engineering design problems," in *AIP conference proceedings*, Vol. 1648 (AIP Publishing LLC, 2015) p. 140010.

[4] C. Coelho, M. F. P. Costa, and L. L. Ferrás, (2023), arXiv:2307.14940 .

[5] C. Coelho, M. F. P. Costa, and L. L. Ferrás, (2023), https://www.kaggle.com/ds/3010437.

[6] C. Coelho, M. F. P. Costa, and L. L. Ferrás, (2023), https://www.kaggle.com/ds/3010478.