

A Filter-Based Neural ODE Approach for Modelling Natural Systems with Prior Knowledge Constraints

C. Coelho[1(✉)], M. Fernanda P. Costa[1], and L.L. Ferrás[1,2]

[1] Centre of Mathematics (CMAT), University of Minho, Braga 4710-057, Portugal
cmartins@cmat.uminho.pt, mfc@math.uminho.pt

[2] Department of Mechanical Engineering (Section of Mathematics) - FEUP, University of Porto, Porto 4200-465, Portugal
lferras@fe.up.pt

Abstract. Neural Ordinary Differential Equations (Neural ODEs) have been used to model natural systems because of their continuous dynamics. However, for meaningful predictions, it is critical to accurately extract from the data the underlying rules or laws governing these systems. To overcome the limitations of commonly used penalty methods, we propose a filter-based approach for Neural ODEs that uses a filter methodology to ensure convergence to a feasible and optimal solution for constrained natural systems. The proposed filter methodology prioritises feasible solutions followed by infeasible and non-dominated solutions. The proposed algorithm provides explicit control over the parameter optimisation process, increases the interpretability of the model, and improves the overall performance. We validate the proposed filter-based Neural ODE approach by modelling two different constrained natural systems. Experimental results show that the models trained by our method provide robust generalisation and satisfy the imposed constraints. Our findings indicate that the proposed method represents a promising approach for incorporating prior knowledge constraints into Neural ODE models.

Keywords: Neural ODE · Natural systems · Filter method · Prior knowledge · Interpretability

1 Introduction

Natural systems such as physical, biological, and chemical systems, are often complex and present challenges for predicting their behaviour in transient and dynamic environments. In general, mathematical models that describe the interactions between variables in a system are given by ordinary differential equations (ODEs). However, natural systems are characterised by complex and non-linear relationships between variables that make it difficult to identify their complete underlying dynamics. Therefore, building mathematical models is a difficult task, which leads to models lacking robustness and predictive accuracy [7].

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2025
R. Meo and F. Silvestri (Eds.): ECML PKDD 2023, CCIS 2135, pp. 349–360, 2025.
https://doi.org/10.1007/978-3-031-74633-8_24

Recently, neural networks (NNs) have been used successfully in modelling these systems due to their ability to capture complex non-linear relationships (and patterns) from data [1,7]. Nevertheless, NNs lack the ability to explicitly perceive/incorporate temporal dependencies. To address this limitation, Chen et al. recently proposed the Neural ODE architecture. The Neural ODEs model a continuous-time function (ODE) such that the solution curve closely fits the data [3]. However, natural systems may follow strict rules or constraints (known a priori) that are not verified by the final solution, when solving the governing equations using common numerical methods or common NNs. If the models used to describe these systems do not incorporate these constraints, they may violate the governing laws and may not accurately predict responses to perturbations, leading to incorrect and unmeaningful predictions.

Since NNs are black-box models, there is no guarantee that they can effectively extract the underlying dynamics of a natural system from the training data. This largely depends on the quantity and quality of the training data, and even in the best of cases, there is no guarantee that the exact governing laws are learned by the model [9]. To address this issue, various strategies have been proposed to introduce constraints into NNs [13]. A popular approach involves the introduction of a penalty term, defined by a constraint violation measure, and a fixed penalty parameter ($\mu > 0$), in the loss function. This technique was inspired by well-established penalty methods from non-linear programming [8]. This approach has already been used to incorporate prior knowledge constraints into NNs and Neural ODEs [2,12,14]. However, penalty methods require the choice of an additional hyperparameter μ to enforce the constraints. The value of μ is highly dependent on the specific problem at hand and can significantly affect the convergence rate and quality of the solution, making its choice a tedious and challenging process, that requires experimentation [8]. In the literature, there are other methods to handle constrained problems that avoid the drawbacks associated with penalty methods. Fletcher et al. [8] proposed the filter method that reformulates the constrained problem as a bi-objective optimisation problem. Thus, minimising the constraint violation measure and the objective function simultaneously [8]. In this work, we propose a filter-based Neural ODE approach for modelling natural systems with prior knowledge constraints.

The present paper is structured as follows. Section 2 provides fundamental concepts, including Neural ODEs, penalty methods and filter methods. Section 3 presents the proposed filter-based Neural ODE method. Section 4 presents the results on the performance of the proposed method, in which we perform a range of numerical experiments by modelling two natural systems, namely, the world population growth and the evolution of a chemical reaction. The paper ends with the conclusions and some ideas of future research, in Sect. 5.

2 Background

This section provides the background required to comprehend the proposed filter-based Neural ODE method. Thus, we present a brief description of the

Neural ODE architecture and techniques for handling constraints of optimisation problems.

2.1 Neural ODEs

Consider a time series that describes a natural system, $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$, where each $\boldsymbol{x}_i$ is a d dimensional vector at time step i ($i = 1, \ldots, N$). Let $\boldsymbol{Y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N)$ be the corresponding ground-truth output time series, where each $\boldsymbol{y}_i$ is a d^* dimensional response vector at time step i. Let $\hat{\boldsymbol{Y}} = (\hat{\boldsymbol{y}}_1, \hat{\boldsymbol{y}}_2, \ldots, \hat{\boldsymbol{y}}_N)$ be the output response generated by a NN model, where each $\hat{\boldsymbol{y}}_i$ is a d^* dimensional output response vector at time step i.

When modelling time series with NNs, the prediction $\hat{\boldsymbol{y}}_i$ at each time step i is seen as the output of a discrete sequence of layers. Consequently, the NN adjusts a function that maps the input to the output vector, by passing the output of each layer to the next consecutively. This process requires regularly sampled training data, and predictions are only defined in time steps with the same frequency as that used for training [11].

Chen et al. [3] introduced the Neural ODE, an architecture that adjusts a continuous-time function $\boldsymbol{f_\theta}$ whose dynamics is defined by a NN. Predictions are obtained by solving an Initial Value Problem (IVP) with initial condition $(\boldsymbol{y}_0, t_0)$, given by the data vector at the first time step of the training sequence. The ODE is discretised and solved in the time interval (t_0, t_f) using a numerical solver [3]:

$$\{\hat{\boldsymbol{y}}_t\}_{1,\ldots,f} = ODESolve(\boldsymbol{f_\theta}, \boldsymbol{y}_0, [t_0, t_f]),$$

with $\hat{\boldsymbol{y}}_t$ the solution at an arbitrary time step t_i, and t_f the last time step of the training data. To optimise the parameters $\boldsymbol{\theta}$ of a Neural ODE, the Adjoint Sensitivity method is used instead of the traditional backpropagation, for avoiding the high memory cost and additional numerical errors [3].

2.2 Techniques for Handling Constraints

A natural system with prior knowledge constraints can be formulated as a constrained optimisation problem. In general, the mathematical formulation of the problem is

$$\begin{aligned} \underset{\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}}{\text{minimize}} \quad & l(\boldsymbol{\theta}) \\ \text{subject to} \quad & \boldsymbol{c}_i(\boldsymbol{\theta}) = 0, \;\; i \in \varepsilon, \\ & \boldsymbol{c}_i(\boldsymbol{\theta}) \leq 0, \;\; j \in \mathcal{I} \end{aligned} \tag{1}$$

where l and $\boldsymbol{c}_i$ are the real-valued objective and constraint functions, on a subset of $\mathbb{R}^{n_\theta}$ ($i \in \varepsilon \cup \mathcal{I}$), with ε and $\mathcal{I}$ the index sets of inequality and equality constraints, respectively. The set of points that satisfy all the constraints define the feasible set $\mathcal{S} = \{\boldsymbol{\theta} \in \mathbb{R}^{n_\theta} : \boldsymbol{c}_i(\boldsymbol{\theta}) = 0, \, i \in \varepsilon; \; \boldsymbol{c}_i(\boldsymbol{\theta}) \leq 0, \, i \in \mathcal{I}\}$.

In the literature, there exist many approaches for solving constrained optimisation problems. The most popular methods are penalty methods that transform constrained problems into unconstrained ones by minimising a penalty function. The penalty function is defined by the objective function plus additional penalty terms (constraint violation measures) multiplied by a positive penalty parameter μ. The goal of the penalty terms is to penalise the objective function l when a trial solution $\tilde{\boldsymbol{\theta}}$ is infeasible, i.e., when $\tilde{\boldsymbol{\theta}} \notin \mathcal{S}$. The mathematical formulation of the unconstrained problem is

$$\underset{\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}}{\text{minimize}} \quad l(\boldsymbol{\theta}) + \mu P(\boldsymbol{\theta}), \tag{2}$$

where $P(\boldsymbol{\theta})$ is the penalty term and $\mu > 0$ is the penalty parameter.

The penalty parameter aims to balance the objective function l and constraints violation P. However, the choice of an appropriate initial value and the subsequent tuning of its values, during the iterative process, are critical since the performance of the algorithm is affected by their values. In some cases, the optimal solution of problem (2) can only be achieved when the penalty parameter goes to infinity. With exact penalty functions, there exists a positive threshold μ^* such that for all $\mu > \mu^*$, the optimal solution of the problem (1) is an optimal solution of the penalty function (unconstrained problem) [10].

The most widely used penalty functions are the quadratic penalty function, the L1 exact penalty function, and the augmented Lagrangian function [10]. For exact penalty and augmented Lagrangian functions, there exists a positive threshold such that any penalty parameter above it guarantees the unconstrained penalty function has the same optimal solutions as the original constrained problem. However, selecting and adjusting the penalty parameter during iterations is crucial for performance. If the parameter μ is too small, the penalty function may become unbounded below, causing the iterates to diverge even if a solution exists. Practical implementations should include mechanisms to increase μ (and reset the starting point if necessary) when constraint violations decrease too slowly or divergence is observed. In practice, the initial value of μ often requires problem-specific knowledge, experimentation, and fine-tuning [10].

To avoid the drawbacks of penalty methods, Fletcher et al. [8] proposed the filter method, which uses the domination concept of multi-objective optimisation. The filter method keeps the two goals separately, minimising the objective function l and the constraint violations P,

$$\underset{\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}}{\text{minimize}} \quad (P(\boldsymbol{\theta}), l(\boldsymbol{\theta})). \tag{3}$$

The filter method accepts a trial point $\tilde{\boldsymbol{\theta}}$ as a new point if the pair $(P(\tilde{\boldsymbol{\theta}}), l(\tilde{\boldsymbol{\theta}}))$ is not *dominated* by any previous pair $(P(\boldsymbol{\theta}_m), l(\boldsymbol{\theta}_m))$ generated by the algorithm. According to [8] we have the following:

1. A pair $(P(\boldsymbol{\theta}_m), l(\boldsymbol{\theta}_m))$ dominates another pair $(P(\boldsymbol{\theta}_t), l(\boldsymbol{\theta}_t))$ if both

$$P(\boldsymbol{\theta}_m) \leq P(\boldsymbol{\theta}_t) \text{ and } l(\boldsymbol{\theta}_m) \leq l(\boldsymbol{\theta}_t),$$

with at least one inequality being strict.

2. A filter $\mathcal{F}$ is a list of pairs $(P(\boldsymbol{\theta}_m), l(\boldsymbol{\theta}_m))$ such that no pair dominates another pair.
3. A point $\tilde{\boldsymbol{\theta}}$ is accepted by the filter if its pair $(P(\tilde{\boldsymbol{\theta}}), l(\tilde{\boldsymbol{\theta}}))$ is not dominated by any pair in the filter.
4. When a point $\tilde{\boldsymbol{\theta}}$ is accepted by the filter, all dominated points are removed from the filter.

The filter method exhibits slightly stronger convergence properties when an envelope is added to the filter [4]. According to [4], the acceptability criterion is modified by imposing a sufficient decrease condition.
A trial point $\tilde{\boldsymbol{\theta}}$ is acceptable to the filter if, for all pairs $(P(\boldsymbol{\theta}_m), l(\boldsymbol{\theta}_m))$ in the filter, the following condition holds

$$P(\tilde{\boldsymbol{\theta}}) \leq (1-\beta)P(\boldsymbol{\theta}_m) \;\text{ or }\; l(\tilde{\boldsymbol{\theta}}) \leq l(\boldsymbol{\theta}_m) - \beta P(\tilde{\boldsymbol{\theta}})$$

with $\beta \in (0,1)$. In practise, $\beta = 10^{-5}$ is commonly used.

If the constraint violation reaches a sufficiently small value, only a sufficient reduction is imposed on function l

$$l(\tilde{\boldsymbol{\theta}}) \leq l(\boldsymbol{\theta}) - \alpha\Delta$$

where α is the step size and $\Delta > 0$ tends to zero throughout the iterative process.

In the following section, we describe how our algorithm integrates some ideas from the filter methodology.

3 Filter-Based Neural ODE Method

Our method follows the ideas of the filter methodology, optimising the constraints violation P and the objective function l simultaneously. As mentioned above, the goal of our method is to incorporate prior knowledge, in the form of constraints, into Neural ODE models. The use of the filter methodology aims to guarantee sufficient progress towards feasible and optimal solutions of constrained natural systems.

In the context of the bi-objective problem (3), to extend the Neural ODE in the sense that both P and l are to be minimised, two optimisers are used. One optimiser to miminise P and other to minimise l, both with respect to $\boldsymbol{\theta}$. Thus, instead of computing a single update of NN parameters during each iteration, as in traditional NN training, our algorithm performs two independent parameter computations.

The filter-based Neural ODE algorithm generates, at the current iteration, two trial points. The trial point $\tilde{\boldsymbol{\theta}}_l$ generated by the optimiser when minimising l, and the trial $\tilde{\boldsymbol{\theta}}_P$ when minimising P. Then, the corresponding pairs $(P(\tilde{\boldsymbol{\theta}}_l), l(\tilde{\boldsymbol{\theta}}_l))$, $(P(\tilde{\boldsymbol{\theta}}_P), l(\tilde{\boldsymbol{\theta}}_P))$ are evaluated and the filter is updated.

In our filter method, the identification of infeasible non-dominated points is based only on the dominance criterion. It also imposes an upper bound $P_{\max}$ on the acceptable constraint violation, i.e., whenever $P(\tilde{\boldsymbol{\theta}}) > P_{\max}$ the point $\tilde{\boldsymbol{\theta}}$ is not accepted by the filter. Moreover, feasible points are not added to the filter to prevent the 'blocking entries' problem from occurring (in particular, of infeasible

points that are very close to the feasibility region). Note that, if a feasible point $\boldsymbol{\theta}^F$ was added to the filter, any infeasible point $\boldsymbol{\theta}^I$ near to the feasibility region and with value of $l(\boldsymbol{\theta}^I)$ greater than the value of $l(\boldsymbol{\theta}^F)$, would be rejected.

Furthermore, whenever the number of points in the filter exceeds a maximum number $N_{\max}$ allowed, the filter is resized by keeping the $N_{\max}$ pairs with lower P values and the upper bound $P_{\max}$ is updated by

$$P_{\max} = \upsilon \max\{P(\boldsymbol{\theta}_m), \text{ for all } (P(\boldsymbol{\theta}_m), l(\boldsymbol{\theta}_m)) \text{ in the filter}\}$$

where $\upsilon > 1$.

For selecting between both trial points, the filter-based Neural ODE algorithm uses the following definition of preference point:

Definition 1. *The rules to define the preference point are the following:*

- *if both trial points are feasible, the trial point with the lowest l value is selected for the next point;*
- *if one trial point is feasible and the other is infeasible, the feasible point is selected;*
- *if both trial points are infeasible and non-dominated, the trial point with lowest P value is selected;*
- *if both trial points are infeasible and only one is non-dominated, the non-dominated is selected;*
- *if both trial points are infeasible and dominated, the current point is kept.*

4 Experimental Evaluation

To analyse the performance of the proposed filter-based Neural ODE method, two natural systems defined by two synthetic datasets were used. For each dataset, three distinct tasks were performed to test and validate the models trained with the proposed method: (a) reconstruction of the data (the same data for training and testing); (b) extrapolation to a longer time horizon (testing in a larger time interval); (c) completion (higher sampling frequency in the test set). The reconstruction task assesses how well the models could capture the underlying dynamics of the system and reproduce the observed data. The extrapolation task evaluates the models' ability to capture the long-term behaviour of the system and make accurate predictions for future time points. The objective of the completion task is evaluating the models' ability to handle datasets with missing or incomplete data, which is commonly encountered in real-life scenarios. The objective is to assess how well the models can fill in the missing data and accurately complete the time series at the higher sampling frequency. The evaluation was performed by computing the Mean Squared Error (MSE) and constraints violation P. As a baseline, we also evaluated the performance of "traditional" Neural ODEs as proposed in [3]. In our experiments, the loss function l is the MSE and the constraints violation P is defined by

$$P(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i\in\varepsilon} |c_i(\boldsymbol{\theta})|^2 + \frac{1}{N}\sum_{i\in\mathcal{I}} (\max(0, c_i(\boldsymbol{\theta})))^2.$$

Table 1 and Table 2 show, for each dataset, the average value of the MSE (MSE_{avg}) and the average value of the constraints violation (P_{avg}), obtained in three independent runs, as well as the standard deviation (std). In the experiments we set $k_{\max} = 10000$, $\upsilon = 10$, $P_{\max} = 100$, $N_{\max} = 20$ for the filter-based Neural ODEs. Both architectures used Adam with learning rate $1e-5$ and 1000 iterations were done to train Neural ODE.

4.1 World Population Growth

The World Population Growth (WPG) dataset can be downloaded from *Kaggle* and has two features namely, time-step t and population $P(t)$. It is characterised by having an upper bound constraint, the carrying capacity, $P(t) \leq 12$ that is satisfied by all data points and should be satisfied by the adjusted models [6]. The WPG dataset was modelled using both Neural ODE and filter-based Neural ODE, employing the same neural network (NN) architecture and training conditions. The NN architecture consists of four hidden layers: a linear layer with 50 neurons, followed by a hyperbolic tangent (tanh) activation function, another linear layer with 50 neurons, and finally an Exponential Linear Unit (ELU) activation function. The input and output layers of the NN consist of a single neuron each.

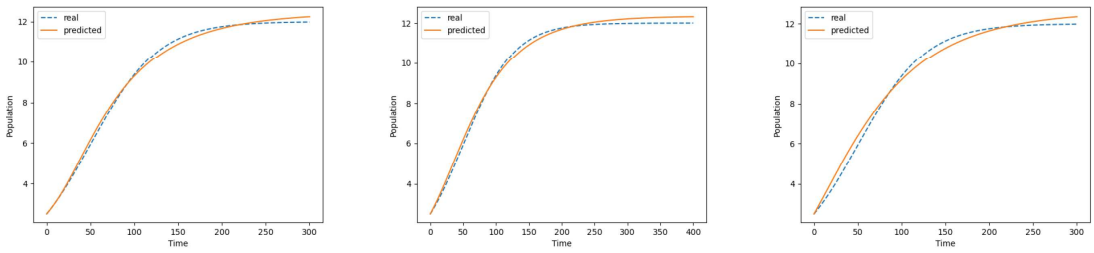
The ground-truth and predicted curves are plotted in Fig. 1 and Fig. 2, for each task, obtained by the best model.

Table 1. Neural ODE and filter-based Neural ODE performance on WPG dataset.

Experiment	Neural ODE		Filter-based Neural ODE	
	MSE_{avg} ± std	P_{avg} ± std	MSE_{avg} ± std	P_{avg} ± std
Reconstruction	0.18672 ± 0.1534	0.07766 ± 0.05142	**0.01769 ± 0.02105**	**0.0014 ± 0.00196**
Extrapolation	0.12771 ± 0.07157	0.15453 ± 0.05251	**0.00225 ± 0.00057**	**0.00854 ± 0.00311**
Completion	0.04292 ± 0.02959	0.02805 ± 0.01166	**0.00238 ± 0.00037**	**0.00012 ± 0.00011**

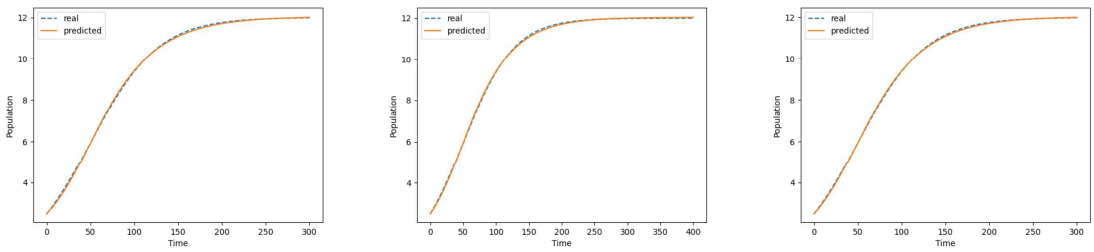
Models trained using our method show lower MSE values and lower violation values P, Table 1. We can conclude that the upper bound limit was successfully introduced into the adjusted filter-based Neural ODE models, while Neural ODE models were not capable of achieving this, Figs. 1 and 2. The superior predictive capabilities of our method can be seen in the extrapolation task, which highlights its remarkable capacity to generalise beyond the training data to unseen future time horizons. Note that although the extrapolation task is conceptually more challenging as the model predicts unseen time-steps, it yielded the best MSE value for the filter-based Neural ODE in our experiments. By examining the data dynamics shown in Fig. 2, it becomes evident that the extrapolation points closely adhere to the upper bound limit constraint. This clearly demonstrates the advantage of explicitly incorporating the constraint into the model.

It is noteworthy that the filter-based Neural ODE exhibits lower MSE values compared to the standard Neural ODE, which may seem counter-intuitive at first. One might expect the Neural ODE to yield the lowest MSE values since it focuses solely on minimising the objective function, while the filter-based Neural ODE tackles the more challenging task of minimising the objective function while satisfying the constraints. However, the explicit incorporation of constraints in the filter-based Neural ODE introduces additional information from the training data, providing a shortcut for learning the system dynamics. This constraint-aware approach enables the model to not only achieve lower constraint violations, as anticipated, but also better fit the observed data. Furthermore, the preference point strategy contributes to directing the search for optimal parameters in a more controlled manner. Unlike traditional neural network training used by Neural ODEs, this strategy prevents the solutions from deteriorating over time, while the Adam optimiser introduces an element of randomness in the optimisation process.



(a) (b) (c)

Fig. 1. Plot of the real (dashed line) and predicted (solid line) curves, for the WPG, on different tasks trained with a Neural ODE: (a) Reconstruction; (b) Extrapolation; (c) Completion.



(a) (b) (c)

Fig. 2. Plot of the real (dashed line) and predicted (solid line) curves, for the WPG, on different tasks trained with the filter-based Neural ODE method: (a) Reconstruction; (b) Extrapolation; (c) Completion.

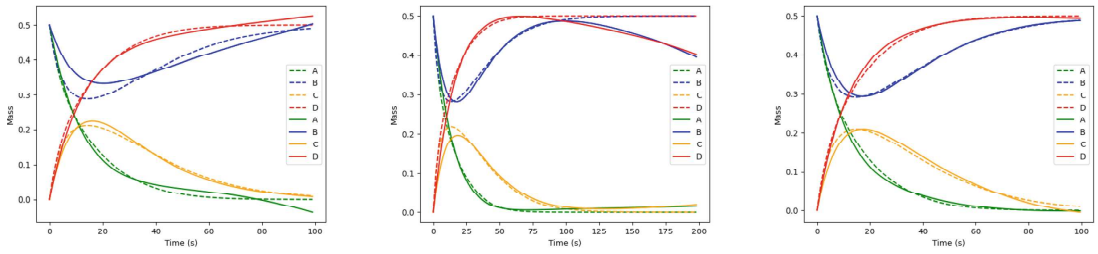
4.2 Chemical Reaction

The Chemical Reaction (CR) dataset can be downloaded from *Kaggle* has five features namely, time-step t and the masses of four chemical species A, B, C, D. It is characterised by having an equality constraint, the conservation of mass, $A(t) + B(t) + C(t) + D(t) = constant$ that is satisfied by all data points and should be satisfied by the adjusted models [5]. The CR dataset was modelled using both Neural ODE and filter-based Neural ODE, employing the same neural network (NN) architecture and training conditions. The NN architecture consists of six hidden layers: a linear layer with 50 neurons, followed by a hyperbolic tangent (tanh) activation function, another linear layer with 64 neurons, and an Exponential Linear Unit (ELU) activation function. This is followed by another linear layer with 50 neurons, and tanh activation function. The input and output layers have 4 neurons each. The ground-truth and predicted curves are plotted in Fig. 3 and Fig. 4, for each task, obtained by the best model.

Table 2. Neural ODE and filter-based Neural ODE performance on the CR dataset.

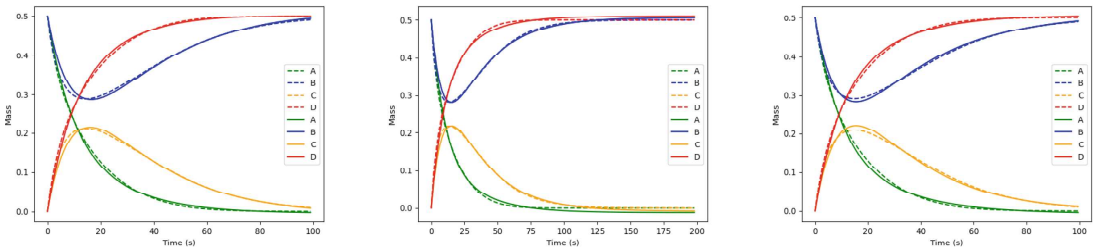
Experiment	Neural ODE		Filter-based Neural ODE	
	MSE_{avg} ± std	P_{avg} ± std	MSE_{avg} ± std	P_{avg} ± std
Reconstruction	0.00211 ± 0.00147	0.00775 ± 0.00669	**0.00069 ± 0.00074**	**7.6667e-06 ± 5.7927e-06**
Extrapolation	0.04114 ± 0.0572	0.01633 ± 0.01644	**0.03257 ± 0.04591**	**0.00703 ± 0.00989**
Completion	0.0009 ± 0.00059	0.00264 ± 0.00307	**0.00032 ± 0.00035**	**7.6666e-06 ± 1.0143e-05**

The performance of the models trained using our method shows lower values of MSE and much lower constraints violation P, Table 2. These indicate that the models obtained using the filter-based Neural ODE method satisfy the law of the conservation of mass and fit the data, making its predictions meaningful. Figure 3 and Fig. 4 demonstrate the efficacy of the proposed method in comparison to Neural ODEs. Similar to the results observed in the WPG dataset, the filter-based Neural ODE demonstrates lower MSE values compared to Neural ODE in modelling the CR dataset. This behaviour can be attributed to the explicit



(a) (b) (c)

Fig. 3. Plot of the real curves (dashed line) and predicted curves (solid line), for the CR dataset, on different tasks trained with a Neural ODE: (a) Reconstruction; (b) Extrapolation; (c) Completion.



(a) (b) (c)

Fig. 4. Plot of the real curves (dashed line) and predicted curves (solid line), for the CR dataset, on different tasks trained with the filter-based Neural ODE method: (a) Reconstruction; (b) Extrapolation; (c) Completion.

incorporation of constraints in the filter-based Neural ODE, which introduces additional information from the training data. Also, the preference point strategy employed by the filter-based Neural ODE directs the optimisation process in a way that aims to prevent the deterioration of both the objective function and constraint violation values. This is a distinct characteristic not present in the training process of traditional Neural ODEs.

5 Conclusion

In this work, a filter-based Neural ODE method that models natural systems and incorporates prior knowledge constraints is proposed.

With the filter method, we aim to minimise the constraints violation and the loss function simultaneously. At each iteration, the filter-based Neural ODE method generates two independent trial points. One computed by minimising the loss function and another minimising the constraints violation. The proposed filter-based Neural ODE method prioritises feasible solutions, followed by infeasible and non-dominated ones.

To assess the effectiveness of the filter-based Neural ODE method, two natural systems with constraints were modelled and evaluated across three distinct tasks (reconstruction, extrapolation and completion). Performance was analysed using the MSE and constraints violation. The results demonstrate that models trained using our newly proposed method can fit the data and have the ability to generalise/extrapolate, while accounting for the natural system constraints.

The proposed method allows the explicit introduction of prior knowledge constraints, crucial when modelling systems that follow governing laws, improving the interpretability of the models obtained by incorporating domain-specific knowledge and ensuring that the model adheres to the known constraints of the system.

The improved fit to the data achieved by the filter-based Neural ODE highlights the significance of incorporating domain-specific constraints in the mode ling process. By leveraging the constraints and utilising the additional information they provide, the filter-based Neural ODE can better capture the true

dynamics of the system. Consequently, it outperforms the standard Neural ODE in terms of both constraint satisfaction and data fitting, demonstrating the benefits of explicitly considering constraints when modelling complex systems.

The filter-based methodology proposed in our method can be extended to any NN architecture and it is a promising approach for the integration of prior knowledge, in the form of constraints into NNs.

Acknowledgements. The authors acknowledge the funding by Fundação para a Ciência e Tecnologia (Portuguese Foundation for Science and Technology) through CMAT projects UIDB/00013/2020 and UIDP/00013/2020 and the funding by FCT and Google Cloud partnership through projects CPCA-IAC/AV/589164/2023 and CPCA-IAC/AF/589140/2023.
C. Coelho would like to thank FCT for the funding through the scholarship with reference 2021.05201.BD.

This work is also financially supported by national funds through the FCT/MCTES (PIDDAC), under the project 2022.06672.PTDC - iMAD - Improving the Modelling of Anomalous Diffusion and Viscoelasticity: solutions to industrial problems.

References

1. Atz, K., Grisoni, F., Schneider, G.: Geometric deep learning on molecular representations. Nat. Mach. Intell. **3**(12), 1023–1032 (2021)
2. Carreira-Perpinan, M.A., Idelbayev, Y.: Learning-Compression algorithms for neural net pruning. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8532–8541. IEEE, Salt Lake City, UT (2018). https://doi.org/10.1109/CVPR.2018.00890
3. Chen, R.T., Rubanova, Y., Bettencourt, J., Duvenaud, D.K.: Neural ordinary differential equations. In: Advances in Neural Information Processing Systems, vol. 31 (2018)
4. Chin, C.M., Fletcher, R.: On the global convergence of an SLP–filter algorithm that takes EQP steps. Math. Program. **96**(1), 161–177 (2003). https://doi.org/10.1007/s10107-003-0378-6
5. Coelho, C., Costa, M.F.P., Ferrás, L.L.: Synthetic chemical reaction (2023). https://doi.org/10.34740/KAGGLE/DS/3010478, https://www.kaggle.com/ds/3010478
6. Coelho, C., Costa, M.F.P., Ferrás, L.L.: World population growth (2023). https://doi.org/10.34740/KAGGLE/DS/3010437, https://www.kaggle.com/ds/3010437
7. Faroughi, S.A., Roriz, A.I., Fernandes, C.: A meta-model to predict the drag coefficient of a particle translating in viscoelastic fluids: a machine learning approach. Polymers **14**(3), 430 (2022). https://doi.org/10.3390/polym14030430
8. Fletcher, R., Leyffer, S., Toint, P.L., et al.: A brief history of filter methods. Preprint ANL/MCS-P1372-0906, Argonne National Laboratory, Mathematics and Computer Science Division **36** (2006)
9. Greydanus, S., Dzamba, M., Yosinski, J.: Hamiltonian neural networks. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
10. Nocedal, J., Wright, S.J.: Numerical Optimization, 2nd Edition. Springer Series in Operation Research and Financial Engineering, Springer, New York, NY (2006). https://doi.org/10.1007/978-0-387-40065-5

11. Rubanova, Y., Chen, R.T.Q., Duvenaud, D.K.: Latent ordinary differential equations for irregularly-sampled time series. In: Advances in Neural Information Processing Systems, vol. 32. Curran Associates, Inc. (2019)
12. Tuor, A., Drgona, J., Vrabie, D.: Constrained neural ordinary differential equations with stability guarantees (arXiv:2004.10883) (2020)
13. von Rueden, L., et al.: Informed machine learning – a taxonomy and survey of integrating prior knowledge into learning systems. IEEE Trans. Knowl. Data Eng. **35**(1), 614–633 (2023). https://doi.org/10.1109/TKDE.2021.3079836
14. Zhang, S., Constantinides, A.: Lagrange programming neural networks. IEEE Trans. Circ. Syst. II: Analog Digital Signal Process. **39**(7), 441–452 (1992). https://doi.org/10.1109/82.160169