Differential Equations as a Model Prior for Deep Learning and its Applications in Robotics

Michael Lutter & Jan Peters *
Department of Computer Science
Technical University of Darmstadt
Hochschulstr. 10, 64289 Darmstadt, Germany
{Lutter, Peters}@ias.tu-darmstadt.de

1 Introduction

For many decades, much of the scientific knowledge of physics and engineering has been expressed via differential equations. These differential equations describe the underlying phenomena and the relations between different interpretable quantities. Therefore, differential equations are a promising approach to incorporate prior knowledge in machine learning models to obtain robust and interpretable models. Especially, deep networks and differential equations fit naturally as deep networks are differentiable and enable the computation of the partial derivatives in closed form at machine precision (Raissi & Karniadakis, 2018). Therefore, combining deep networks and differential equations is a promising approach to constrain deep networks to learn meaningful representations.

In this paper, we summarize a straight forward approach to incorporate deep networks in differential equations and solve first-order non-linear differential equations by minimising the residual. We describe the deep differential network that computes the functional value and smooth Jacobians in closed form. Afterwards, we summarize two robotics applications that use differential equations as model prior for deep networks to achieve model learning and optimal feedback control. In contrast to prior work, which focused on the specific application, this paper focuses on the technical requirements for deep networks and differential equations and we present open questions that we encountered during our research.

2 Deep Differential Network

To embed a deep network within a first order differential equation and achieve fast training of the network parameters, one requires to compute the Jacobian w.r.t. the network input efficiently. The deep differential network extends the standard feed-forward deep network compute the Jacobian efficiently using a single feed-forward pass. When the softplus activation is used, this network architecture yields good approximations of the Jacobian. The computational graph of the deep differential network (Figure 1) is extended such that each network layer computes the partial derivative w.r.t. the previous layer, i.e. $\partial h_i / \partial h_{i-1}$. Chaining these partial derivatives computes the Jacobian in closed form. Let deep network with N layers model the function $f(\mathbf{x};\theta) : \mathbb{R}^n \to \mathbb{R}^m$ and each layer be an affine transformation of the output of the previous layer h_{i-1} with a subsequent non-linearity g, i.e., $h_i = g(\mathbf{W}_i^T \mathbf{h}_{i-1} + \mathbf{b}_i)$. Then partial derivative of each layer and the Jacobian are described by

$$\frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} = \text{diag}\left(g'(\mathbf{W}_i^T \mathbf{h}_{i-1} + \mathbf{b}_i)\right) \mathbf{W}_i \qquad \frac{\partial f}{\partial \mathbf{x}} = \frac{\partial \mathbf{h}_1}{\partial \mathbf{x}} \frac{\partial \mathbf{h}_2}{\partial \mathbf{h}_1} \cdots \frac{\partial \mathbf{h}_{N-1}}{\partial \mathbf{h}_{N-2}} \frac{\partial \mathbf{y}}{\partial \mathbf{h}_{N-1}}. \tag{1}$$

A PyTorch implementation is available at (non-anonymous repository added after double blind review). We explicitly encode forward-mode automatic differentiation within the feed-forward network as this is favorable compared to the reverse-mode automatic differentiation implemented in major deep learning frameworks including PyTorch and Tensorflow. These frameworks use reverse-mode automatic differentiation as this mode is more efficient for computing the gradient for a scalar loss (i.e., $m \ll n$), which is the main use-case for optimizing deep network parameters. While both modes yield the identical derivatives, forward-mode automatic differentiation is favorable for Jacobians, where $m \approx n$, (Baydin et al., 2017) and this computation requires only a single forward pass through the network while the reverse mode must perform a forward and backward pass. Therefore, this explicit computation is more efficient and this decreased computational complexity is essential to enable real-time control loops required by many robotics applications.

*Max Planck Institute for Intelligent Systems, Spemannstr. 41, 72076 Tübingen, Germany

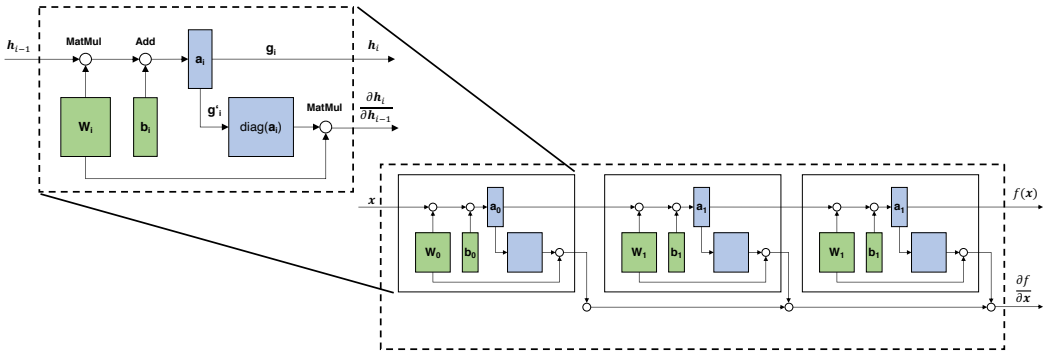


Figure 1: Computational graph of the deep differential network. The standard feed-forward network is extended to compute the Jacobian $\partial f / \partial \mathbf{x}$ in closed form and a single forward pass.

2.1 Evaluating Jacobians

To solve differential equations with a deep differential network, the Jacobian $\partial \mathbf{y} / \partial \mathbf{x}$ must achieve a good approximation of the symbolic expression. This good approximation is not guaranteed as f could be approximated by many step functions and the Jacobians of such representation are undesirable for solving a differential equation. To demonstrate that the Jacobian of the network approximates the true symbolic Jacobian, we train the network to approximate a function and compare the approximated Jacobian to the symbolic Jacobian. In addition, we analyze the effect of different non-linear activations. Figure 2 shows the symbolic and the approximated Jacobian using a deep differential network with relu and softplus activations. The first row shows the true symbolic expression, the second row shows the Jacobian trained using a loss including f and $\partial f / \partial \mathbf{x}$, i.e., $\theta^* = \arg\min (y - f(\mathbf{x};\theta))^2 + (\partial y / \partial \mathbf{x} - \partial f(\mathbf{x};\theta)/\partial \mathbf{x})^2$. The third row corresponds to minimizing the loss only including y, i.e., $\theta^* = \arg\min (y - f(\mathbf{x};\theta))^2$. While both activation approximate f well, only the softplus activation achieves smooth Jacobians that approximate the symbolic expression. The relu activation yields non-smooth Jacobians. Even when trained only on the function f, the Jacobian for the softplus activation matches the symbolic expression. Therefore, softplus activation is preferable compared to the relu non-linearity.

2.2 Solving Differential Equations by Gradient Descent

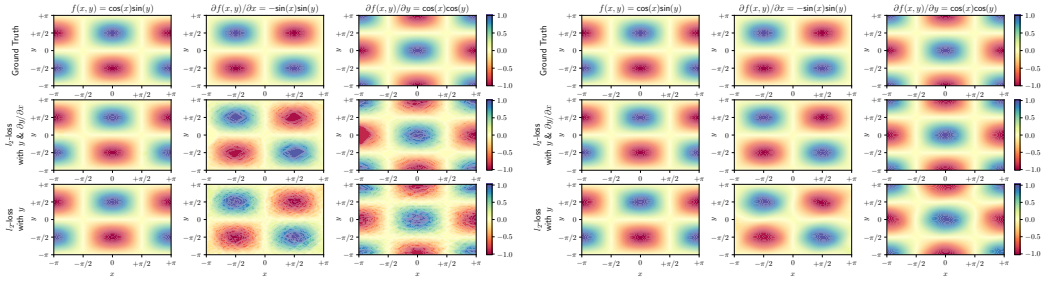
To solve a first order non-linear differential equation without boundary constraint and unique solution, one can simply embed the deep differential network within the differential equation and minimize the residual of the differential equation. In the most general case this minimization problem is described by

$$F\left(\mathbf{x},\, f(\mathbf{x};\theta),\, \frac{\partial f(\mathbf{x};\theta)}{\partial \mathbf{x}}\right) := 0 \qquad \Rightarrow \qquad \theta^* = \arg\min_{\theta} \sum_{i=0}^{N} F\left(\mathbf{x}_i,\, f(\mathbf{x}_i;\theta),\, \frac{\partial f(\mathbf{x}_i;\theta)}{\partial \mathbf{x}}\right)^2 \tag{2}$$

where F is known and x can be sampled on the complete domain Ω. This optimization problem can be solved using the standard gradient based optimization methods of end-to-end deep learning. In section 3 we demonstrate that this approach can be used to obtain solutions to the Euler-Lagrange equation and the Hamilton-Jacobi-Bellman equation.

2.3 Differential Equations with Boundary Constraints

While the previously described approach is straight-forward for problems without boundary constraints, incorporating boundary constraints that make the solution unique is non-trivial. These boundary constraints cannot be simply added as penalty term to Equation 2, as this penalty is only enforced locally. The high capacity deep networks, which are local function approximator, can be attracted to the undesired solution inside the domain and only comply with the boundary constraints within the vicinity of the boundary. Therefore, the learned solution is not coherent across the complete domain and separated by borders of high residual. Furthermore, these discontinuous loss landscapes frequently deteriorate the gradient based optimization. Therefore, incorporating boundary constraints such that a globally coherent solution is learned is an open research question.



(a) (b)

Figure 2: Comparison of the symbolic Jacobian to the approximations of the deep differential network with relu (a) and softplus (b) non-linearity. The second row corresponds to supervised training using y and $\partial y/\partial \mathbf{x}$. The third row corresponds to supervised training using only y. The Jacobian of the differential network with softplus non-linearity approximates the symbolic Jacobian while the relu non-linearity only obtains a discontinuous and coarse approximation.

3 Robotics Applications

In the following, two robotics applications are described. First, the Euler-Lagrange differential equation is solved to obtain physically plausible models. Second, the Hamilton-Jacobi-Bellman differential equation is solved to obtain an optimal policy.

3.1 Euler-Lagrange Equation for Model Learning

To learn models that are physically plausible and interpretable, the Euler-Lagrange differential equation from Lagrangian mechanics can be solved. Embedding two deep differential networks within this equation and minimizing the residual learns a model that is guaranteed to describe a mechanical system with holonomic constraints. Furthermore, this approach enables the unsupervised learning of the interpretable physical forces and energies, i.e., inertial forces, Coriolis forces, gravitational forces as well as potential energy and kinetic energy. Learning these quantities is remarkable as they cannot be directly observed and hence, previous works (Schaal et al., 2002; Nguyen-Tuong & Peters, 2011; Ledezma & Haddadin, 2017; Sanchez-Gonzalez et al., 2018) using supervised learning could not retrieve this information.

The Euler-Lagrange differential equation is described by

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\mathbf{q}}} - \frac{\partial L}{\partial \mathbf{q}} = \boldsymbol{\tau} \tag{3}$$

with the torques $\boldsymbol{\tau}$, the generalized coordinates $\mathbf{q}$, the Lagrangian $L = T - V$, the kinetic energy T and the potential energy V. Representing the potential energy $V(\mathbf{q}; \phi)$ and kinetic energy $T(\mathbf{q}, \dot{\mathbf{q}}; \psi) = \dot{\mathbf{q}}^T \mathbf{H}(\mathbf{q}; \psi)\dot{\mathbf{q}}$ as deep differential networks and enforcing the positive definiteness constraint of $\mathbf{H}(\mathbf{q}; \psi)$ using the Cholesky decomposition, one guarantees that the system dynamics always describe a mechanical system. To obtain the parameters describing the desired mechanical system, the differential equation can be solved by minimizing the residual using stochastic gradient descent on recorded data from the physical system. More concretely, one records a dataset containing $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$ and $\boldsymbol{\tau}$ on the mechanical system and minimizes the squared residual of Equation 3 with SGD to retrieve the energies of the mechanical system.

Figure 3 shows the learned decomposition of the torque signal into the force components as well as the system energies for the Cartpole. The learned model recovers the energies and forces from the super-imposed torque signal and matches the analytic model. Other experiments have shown that this approach of Deep Lagrangian Networks can be applied to real-time energy control (Lutter & Peters, 2019), online learning of non-linear feed-forward tracking control (Lutter et al., 2019b) and optimal control (Gupta et al., 2019).

3.2 Hamilton-Jacobi-Bellman Equation for Optimal Control

To obtain the optimal control policy π^* one can either use reinforcement learning, trajectory optimization or solve the Hamilton-Jacobi-Bellman (HJB) equation when the model is known. For mechanical systems with holonomic constraints and a separable cost function that is strictly convex

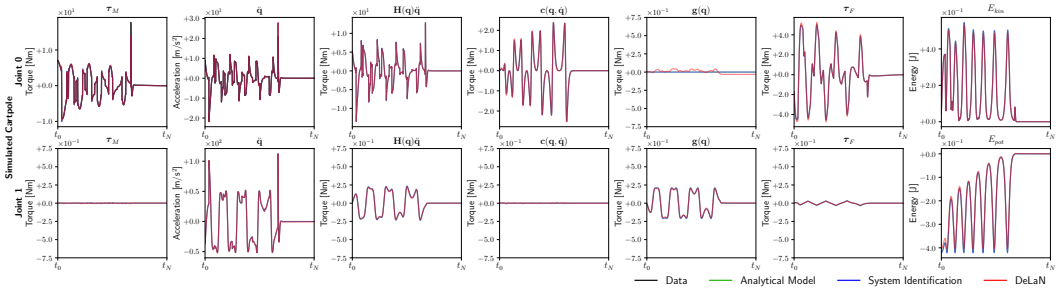


Figure 3: The super-imposed observed torque and the learned decomposition of the inertial-, centrifugal-, Coriolis-, gravitational- and frictional forces as well as the kinetic and potential energy for the swing-up of the simulated Cartpole. The learned physical quantities are learned unsupervised by leveraging the Euler-Lagrange equation as model prior and match the analytic model.

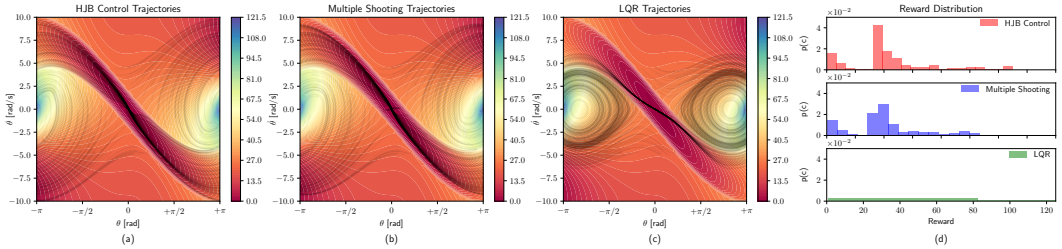


Figure 4: (a-c) Learned value function for the torque-limited pendulum with log-cosine cost and the optimal trajectories for HJB control (a), multiple shooting (b) and LQR (c). (d) Cost distributions $p(c)$ for the sampled starting configurations. The optimal policy obtained by solving the HJB equation with deep differential networks obtains the similar trajectories and cost distribution as multiple shooting.

w.r.t. the actions, the HJB simplifies to a first order non-linear differential equation (Lutter et al., 2019a). The simplified HJB is described by

$$\rho V^*(\mathbf{x}) = q(\mathbf{x}) + \mathbf{a}(\mathbf{x})^T V_{\mathbf{x}}^* - g'(-\mathbf{B}(\mathbf{x})^T V_{\mathbf{x}}^*) \tag{4}$$

with state $\mathbf{x}$, action $\mathbf{u}$, the partial derivative $V_{\mathbf{x}}^* = \partial V^* / \partial \mathbf{x}$, the convex conjugate g', the non-linear continuous time dynamics $\dot{\mathbf{x}} = \mathbf{a}(\mathbf{x}) + \mathbf{B}(\mathbf{x})\mathbf{u}$ and separable cost function $c(\mathbf{x}, \mathbf{u}) = q(\mathbf{x}) + g(\mathbf{u})$. Solving this equation yields the optimal value function and implies the optimal control policy described by $\pi^*(\mathbf{x}) = \nabla g'(-\mathbf{B}(\mathbf{x})^T V_{\mathbf{x}}^*)$. Simply embedding a deep differential network as value function and minimizing the residual is not sufficient to solve this differential equation as the solution is only unique given an additional boundary constraint. To avoid explicitly incorporating the boundary constraint, a homotopy method is used, which anneals the discounting from short sighted, i.e., $\rho = \infty$, to far sighted, i.e., $\rho = 0$. This approach is sufficient as for this specific equation the undesired solution diverges and the desired solution is known, i.e., $V(\mathbf{x}) = 0 \; \forall \; \mathbf{x} \in \Omega$. Therefore, the initial solution can be learned and tracked when annealing ρ.

Figure 4 shows the learned value function of the torque limited pendulum and the applied optimal policy for 300 different starting configurations. The optimal policy obtained by solving the HJB equation achieves comparable performance as the optimal control approach of multiple shooting. In contrast to multiple shooting, the HJB optimal policy does not need to replan for different starting configurations and is a closed-loop policy rather than an open-loop optimal trajectory.

4 Conclusion

In this paper we described deep differential networks and the application to solving first order non-linear differential equations without boundary constraints. In addition, we showed that the deep differential network with softplus non-linearity achieves good approximation of the symbolic Jacobian and discussed the open problem of incorporating boundary constraints. Finally, we summarized two robotics applications that leveraged differential equations as model prior to learn physically plausible models and optimal feedback controllers.

REFERENCES

Atılım Günes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. ***The Journal of Machine Learning Research***, 18(1):5595–5637, 2017.

Jayesh K Gupta, Kunal Menda, Zachary Manchester, and Mykel J Kochenderfer. A general framework for structured learning of mechanical systems. ***arXiv preprint arXiv:1902.08705***, 2019.

Fernando Díaz Ledezma and Sami Haddadin. First-order-principles-based constructive network topologies: An application to robot inverse dynamics. In ***International Conference on Humanoid Robotics (Humanoids), 2017***, pp. 438–445. IEEE, 2017.

M. Lutter and J. Peters. Deep lagrangian networks for end-to-end learning of energy-based control for under-actuated systems. In ***International Conference on Intelligent Robots and Systems (IROS)***, 2019.

Michael Lutter, Boris Belousov, Kim Listmann, Debora Clever, and Jan Peters. Hjb optimal feedback control with deep differential value functions and action constraints. ***Conference on Robot Learning (CoRL)***, 2019a.

Michael Lutter, Christian Ritter, and Jan Peters. Deep lagrangian networks: Using physics as model prior for deep learning. In ***International Conference on Learning Representations (ICLR)***, 2019b.

Duy Nguyen-Tuong and Jan Peters. Model learning for robot control: a survey. ***Cognitive Processing***, 12(4):319–340, 2011.

Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. ***Journal of Computational Physics***, 357:125–141, 2018.

Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. ***arXiv preprint arXiv:1806.01242***, 2018.

Stefan Schaal, Christopher G Atkeson, and Sethu Vijayakumar. Scalable techniques from nonparametric statistics for real time robot learning. ***Applied Intelligence***, 17(1):49–60, 2002.