Sparse-Input Neural Network Augmentations for Differentiable Simulators

Eric Heiden
University of Southern California
heiden@usc.edu

David Millard
University of Southern California
dmillard@usc.edu

Erwin Coumans
Robotics at Google
erwincoumans@google.com

Gaurav S. Sukhatme
University of Southern California
gaurav@usc.edu

Abstract

Differentiable simulators provide an avenue for closing the sim2real gap by enabling the use of efficient, gradient-based optimization algorithms to find the simulation parameters that best fit the observed sensor readings. Nonetheless, these analytical models can only predict the dynamical behavior of systems they have been designed for. In this work, we study the augmentation of a differentiable rigid-body physics engine via neural networks that is able to learn nonlinear relationships between dynamic quantities and can thus learn effects not accounted for in the traditional simulator. By optimizing with a sparse-group lasso penalty, we are able to reduce the neural augmentations to the necessary inputs that are needed in the learned models of the simulator while achieving highly accurate predictions.

1 Introduction

Physics engines enable the accurate prediction of dynamical behavior of systems for which an analytical model has been implemented. Given such models, Bayesian estimation approaches [1] can be used to find the simulation parameters that best fit the real-world observations of the given system. With the advent of differentiable simulators, gradient-based optimizers can further improve the speed of convergence of such inference approaches.

Nonetheless, the sim2real gap remains for most systems we use in the real world. For example, in typical simulations used in robotics, the robot is assumed to be an articulated mechanism with perfectly rigid links, even though they actually bend under heavy loads. Motions are often optimized without accounting for air resistance. In this work, we focus on overcoming such errors due to unmodeled effects by implementing a differentiable rigid-body simulator that is enriched by fine-grained data-driven models.

Our contributions are three-fold:

1. We present an architecture for designing differentiable simulators that enables the augmentation of analytical models with data-driven function approximators at any point in the computation graph. This approach generalizes previous residual and entirely data-driven models for learning dynamics.
2. Through techniques from neural network sparsification, we define an optimization objective that results in minimally invasive augmentations that only depend on input quantities that influence the part of the dynamics being learned.

Workshop on Differentiable Vision, Graphics, and Physics in Machine Learning at NeurIPS 2020.

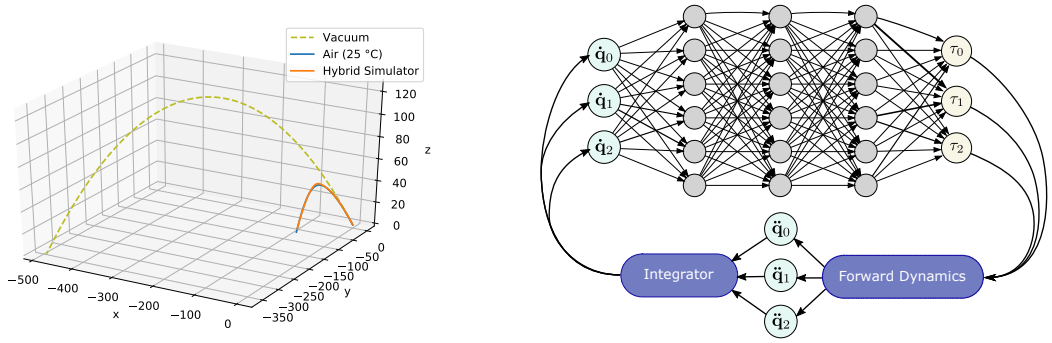

Figure 1: *Left:* trajectories of a golf ball in vacuum and air (at 25 °C temperature). The analytical rigid-body dynamics model of our simulator does not account for air resistance due to the air's density and viscosity. Augmented with a learned model, it can accurately predict the passive forces necessary to match the reference trajectories, generalizing over various golf ball impacts. *Right:* architecture of the hybrid simulator for the golf experiment where a neural network learns passive forces τ given joint velocities $\dot{\mathbf{q}}$ (MLP biases and joint positions $\mathbf{q}$ are not shown).

3. We demonstrate how such identified augmentations can help drastically reduce the model error on example systems involving unmodeled effects, such as joint friction and drag. Given positions and velocities, neural networks that learn corrections on the force level are trained which results in a hybrid simulator that remains compatible with the Newton-Euler equations.

2 Related Work

Various methods learn system dynamics from time series data of a real system. Such "intuitive physics" models often use deep graph neural networks to discover constraints between particles or bodies [2–9]. We propose a general-purpose hybrid simulation approach that combines analytical models of dynamical systems with data-driven residual models that learn parts of the dynamics unaccounted for by the analytical simulation models.

Originating from traditional physics engines [10–12], differentiable simulators have been introduced that leverage automatic, symbolic or implicit differentiation to calculate parameter gradients through the analytical physics models for rigid-body dynamics [13–18], light propagation [19–21], and other physical phenomena [22, 23].

Residual physics models [24–27] augment physics engines with learned models to reduce the sim2real gap, i.e. the error between the simulated and the measured state from the real system. Most of these approaches introduce residual learning to the output of the physics engine, while we propose a more fine-grained approach, similar to Hwangbo et al. [26] and Pfrommer et al. [28], where only a few places inside the simulator are enriched by data-driven models. While the network in [26] learns the actuator dynamics through supervised learning, our end-to-end differentiable model allows backpropagation from indirectly involved quantities to any part of the computation graph, including neural network weights. Such approach has the benefit that quantities, such as a robot's joint angles, are often much easier and more accurately to measure compared to quantities, such as external forces, which are much harder to observe. By differentiating through the loss defined on a trajectory of joint positions and velocities, in this work, we are able to learn passive force models for which accurate training data (on the force level) would be very challenging to obtain.

3 Approach

3.1 Hybrid Simulation

We propose a technique for hybrid simulation that leverages differentiable physics models and neural networks to close the sim2real gap via system identification. By enabling any part of the simulation to be replaced or augmented by neural networks, we can learn unmodeled effects from data. Through

	Analytical	Data-driven	End2end ∇	Hybrid
Physics engine [10–12]	✓			
Residual physics [26, 28, 25, 24, 27]	✓	✓		✓
Learned physics [30, 2, 8, 31]		✓	✓	
Differentiable sim. [13, 22, 14, 15]	✓		✓	
Ours	✓	✓	✓	✓

Table 1: Comparison of dynamics modeling approaches (only selected works) along the axes of analytical and data-driven modeling, end-to-end differentiability, and hybrid approaches.

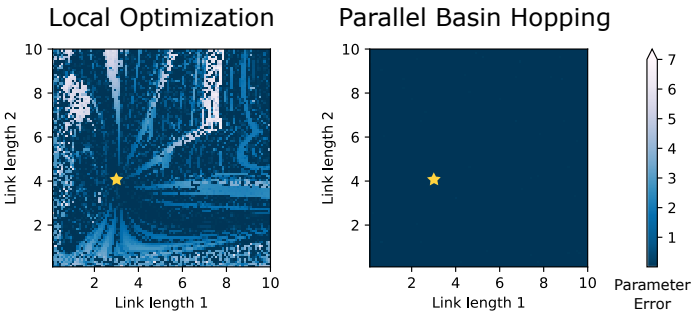

Figure 2: Comparison of system identification results from purely local Levenberg-Marquardt optimization (left) and a random restart strategy (right). The link lengths of a double pendulum are the two simulation parameters to be estimated from joint position trajectories. The true parameters are 3 and 4 (indicated by a star), and the colors indicate the $\ell 2$ distance between the optimized parameters and the ground-truth (darker shade indicates lower error).

template meta-programming, our C++ implementation allows any intermediate value used in a simulation to be augmented by neural networks that accept input connections from any other variable. In our simulation framework, such *neural scalars* are assigned a unique name, to be referenced in a "neural blueprint", which specifies the neural network architecture, and which input and output variables are involved. We compute gradients of the weights and analytical simulation parameters using the multi-dimensional dual number implementation from the Ceres optimizer [29]. Many other automatic differentiation libraries (Stan Math, CppAD, etc.) are supported.

3.2 Overcoming Local Minima

System identification for nonlinear mechanisms, or systems involving contact, results in a highly nonconvex loss landscape which is fraught with poor local minima. Even estimating the link lengths of a double pendulum given a trajectory of joint angles and velocities often results in poor estimates when solely local optimizers, such as Levenberg-Marquardt, are applied (see Figure 2 left). To overcome these problems, we use tools from global optimization, e.g. parallel basin hopping [32], that combine solutions from local optimizers and restart them from many different guesses. We further leverage Pagmo [33] that provides a library of population-based algorithms and other global optimization approaches that run many gradient-based solvers in parallel.

4 Experiments

4.1 Learning Air Resistance in Golfing

To demonstrate how differentiable simulators can benefit from neural networks augmenting the implemented dynamics equations, we equip a rigid-body physics engine with a learned model that accounts for drag. Such phenomenon has nonlinear effects on the dynamics of objects moving through a medium, such as air, that has nonzero viscosity and density. Our hybrid simulator equipped with a three-layer feed-forward neural network (Figure 1 right) learns the passive forces necessary to

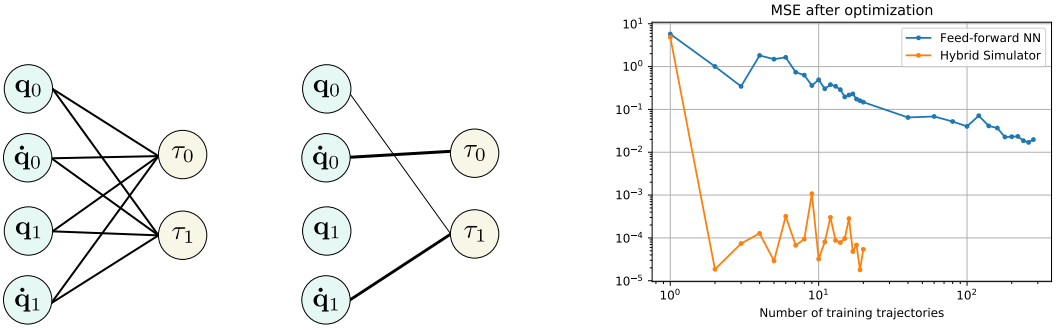

Figure 3: *Left:* Neural augmentation for a double pendulum that learns joint damping which is a force τ_i that linearly depends on $\dot{\mathbf{q}}_i$. Given the fully connected neural network (left), after 15 optimization steps using the sparse group lasso cost function (Equation 1) (right), the input layer (light green) is sparsified to only the relevant quantities that influence the dynamics of the observed system. *Right:* Final mean squared error (MSE) on test data after training a feed-forward neural network and our hybrid physics engine on a set of trajectories from a double pendulum that has joint friction and stiffness. While the neural-network baseline is able to closely match the dynamics after a few hundred trajectories, our model achieves a higher accuracy with orders of magnitude less training samples.

correct the analytical physics engine that models the golf ball's motion in vacuum. Trained on 20 different golf ball pushes (forces), the hybrid simulator generalizes well to other initial conditions.

4.2 Discovering Neural Augmentations

In our previous experiments we carefully selected which simulation quantities to augment by neural networks. While it is expected that air resistance and other effects result in passive forces that need to be applied to the rigid bodies in the simulator, it is often difficult to foresee which dynamical quantities actually influence the dynamics of the observed system. Since we aim to find minimally invasive data-driven models, we focus on neural network augmentations that have the least possible effect while achieving the goal of high predictive accuracy. Inspired by Sparse Input Neural Networks (SPINN) [34], we adapt the sparse-group lasso [35] penalty for the cost function:

$$\mathcal{L} = \sum_t ||f_\theta(s_{t-1}) - s_t^*||_2^2 + \kappa||\theta_{[:1]}||_1 + \lambda||\theta_{[1:]}||_2^2, \tag{1}$$

given weighting coefficients $\kappa, \lambda > 0$, where $f_\theta(s_{t-1})$ is the state predicted by the hybrid simulator given the previous simulation state s_{t-1} (consisting of $\mathbf{q}, \dot{\mathbf{q}}$), s_t^* is the state from the observed system, $\theta_{[:1]}$ are the network weights of the first layer, and $\theta_{[1:]}$ are the weights of the upper layers.

We demonstrate the approach on a double pendulum system that is modeled as a frictionless mechanism in our analytical physics engine. The reference system, on the other hand, is assumed to exhibit joint friction, i.e. a force proportional to and opposed to the joint velocity. As expected, compared to a fully learned dynamics model, our hybrid simulator outperforms the baselines significantly (Figure 3). As shown on the right, convergence is achieved while requiring orders of magnitude less training samples. At the same time, the resulting neural network augmentation converges to weights that clearly show that the joint velocity $\dot{\mathbf{q}}_i$ influences the residual joint force τ_i that explains the observed effects of joint friction in joint i (Figure 3 left).

5 Conclusion

We presented a differentiable simulator for articulated rigid-body dynamics that allows the insertion of neural networks at any point in the computation graph. Through gradient-based optimization, nonlinear effects can be learned by these data-driven models while their contributions are kept at a minimum through the sparse-group lasso penalty. Based on our results on more complex systems [36], we further plan to investigate real-world applications in robotics, where the sim2real gap can be reduced by leveraging our hybrid simulation approach.

References

[1] Fabio Ramos, Rafael Carvalhaes Possas, and Dieter Fox. Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators. *Robotics: Science and Systems*, 2019.

[2] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems*, pages 4502–4510, 2016.

[3] Zhenjia Xu, Jiajun Wu, Andy Zeng, Joshua B Tenenbaum, and Shuran Song. DensePhysNet: Learning dense physical object representations via multi-step dynamic interactions. *Robotics: Science and Systems*, June 2019.

[4] Siyu He, Yin Li, Yu Feng, Shirley Ho, Siamak Ravanbakhsh, Wei Chen, and Barnabás Póczos. Learning to predict the cosmological structure formation. *Proceedings of the National Academy of Sciences of the United States of America*, 116(28):13825–13832, July 2019.

[5] Maziar Raissi, Hessam Babaee, and Peyman Givi. Deep learning of turbulent scalar mixing. Technical report, 2018.

[6] Damian Mrowca, Chengxu Zhuang, Elias Wang, Nick Haber, Li Fei-Fei, Joshua B Tenenbaum, and Daniel L K Yamins. Flexible neural representation for physics prediction. *Advances in Neural Information Processing Systems*, June 2018.

[7] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pages 6571–6583, 2018.

[8] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B. Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rJgbSn09Ym`.

[9] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks, 2020.

[10] Erwin Coumans et al. Bullet physics library. *Open source: bulletphysics.org*, 15(49):5, 2013.

[11] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[12] Jeongseok Lee, Michael X. Grey, Sehoon Ha, Tobias Kunz, Sumit Jain, Yuting Ye, Siddhartha S. Srinivasa, Mike Stilman, and C. Karen Liu. Dart: Dynamic animation and robotics toolkit. *Journal of Open Source Software*, 3(22):500, 2018. doi: 10.21105/joss.00500. URL `https://doi.org/10.21105/joss.00500`.

[13] Markus Giftthaler, Michael Neunert, Markus Stäuble, Marco Frigerio, Claudio Semini, and Jonas Buchli. Automatic differentiation of rigid body dynamics for optimal control and estimation. *Advanced Robotics*, 31(22):1225–1237, 2017. doi: 10.1080/01691864.2017.1395361.

[14] Justin Carpentier and Nicolas Mansard. Analytical derivatives of rigid body dynamics algorithms. In *Robotics: Science and Systems*, 2018.

[15] Filipe de Avila Belbute-Peres, Kevin Smith, Kelsey Allen, Josh Tenenbaum, and J. Zico Kolter. End-to-end differentiable physics for learning and control. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7178–7189. Curran Associates, Inc., 2018. URL `http://papers.nips.cc/paper/7948-end-to-end-differentiable-physics-for-learning-and-control.pdf`.

[16] Twan Koolen and Robin Deits. Julia for robotics: simulation and real-time control in a high-level programming language. In *International Conference on Robotics and Automation*, 05 2019.

[17] Eric Heiden, David Millard, Hejia Zhang, and Gaurav S. Sukhatme. Interactive differentiable simulation. *CoRR*, abs/1905.10706, 2019. URL `http://arxiv.org/abs/1905.10706`.

[18] Eric Heiden, David Millard, and Gaurav S. Sukhatme. Real2sim transfer using differentiable physics. *R:SS Workshop on Closing the Reality Gap in Sim2real Transfer for Robotic Manipulation*, 2019.

[19] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: A retargetable forward and inverse renderer. *Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 38 (6), November 2019. doi: 10.1145/3355089.3356498.

[20] Eric Heiden, Ziang Liu, Ragesh K. Ramachandran, and Gaurav S. Sukhatme. Physics-based simulation of continuous-wave LIDAR for localization, calibration and tracking. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2020.

[21] Jatavallabhula Krishna Murthy, Ganesh Iyer, and Liam Paull. gradslam: Dense slam meets automatic differentiation. In *International Conference on Robotics and Automation (ICRA)*, 2020.

[22] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. DiffTaichi: Differentiable programming for physical simulation. *ICLR*, 2020.

[23] Junbang Liang, Ming Lin, and Vladlen Koltun. Differentiable cloth simulation for inverse problems. In *Advances in Neural Information Processing Systems*, pages 771–780, 2019.

[24] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. 2019.

[25] Anurag Ajay, Jiajun Wu, Nima Fazeli, Maria Bauza, Leslie P Kaelbling, Joshua B Tenenbaum, and Alberto Rodriguez. Augmenting Physical Simulators with Stochastic Neural Networks: Case Study of Planar Pushing and Bouncing. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[26] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

[27] Florian Golemo, Adrien Ali Taiga, Aaron Courville, and Pierre-Yves Oudeyer. Sim-to-real transfer with neural-augmented robot simulation. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 817–828. PMLR, 29–31 Oct 2018. URL `http://proceedings.mlr.press/v87/golemo18a.html`.

[28] Samuel Pfrommer, Mathew Halm, and Michael Posa. Contactnets: Learning of discontinuous contact dynamics with smooth, implicit representations, 2020.

[29] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. `http://ceres-solver.org`.

[30] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W Battaglia. Learning to simulate complex physics with graph networks. *arXiv preprint arXiv:2002.09405*, 2020.

[31] Yifeng Jiang and C. Karen Liu. Data-augmented contact model for rigid body simulation. *CoRR*, abs/1803.04019, 2018. URL `http://arxiv.org/abs/1803.04019`.

[32] Steven L McCarty, Laura M Burke, and Melissa McGuire. Parallel monotonic basin hopping for low thrust trajectory optimization. In *2018 Space Flight Mechanics Meeting*, page 1452, 2018.

[33] Francesco Biscani and Dario Izzo. A parallel global multiobjective framework for optimization: pagmo. *Journal of Open Source Software*, 5(53):2338, 2020. doi: 10.21105/joss.02338. URL `https://doi.org/10.21105/joss.02338`.

[34] Jean Feng and Noah Simon. Sparse-input neural networks for high-dimensional nonparametric regression and classification, 2019.

[35] Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A sparse-group lasso. *Journal of computational and graphical statistics*, 22(2):231–245, 2013.

[36] Eric Heiden, David Millard, Erwin Coumans, Yizhou Sheng, and Gaurav S. Sukhatme. NeuralSim: Augmenting differentiable simulators with neural networks. *arXiv preprint*, 2020.