Augmenting Differentiable Simulators with Neural Networks to Close the Sim2Real Gap

Eric Heiden*, David Millard*, Erwin Coumans†, and Gaurav S. Sukhatme*
*Robotics Embedded Systems Lab
University of Southern California, Los Angeles, California 90089
Email: {heiden,dmillard,gaurav}@usc.edu
†Google, Mountain View, California 94043
Email: erwincoumans@google.com

***Abstract*—We present a differentiable simulation architecture for articulated rigid-body dynamics that enables the augmentation of analytical models with neural networks at any point of the computation. Through gradient-based optimization, identification of the simulation parameters and network weights is performed efficiently in preliminary experiments on a real-world dataset and in sim2sim transfer applications, while poor local optima are overcome through a random search approach.**

I. Introduction and Related Work

Simulators are crucial tools for planning and control algorithms to tackle difficult real world robotics problems. In many cases, however, such models diverge from reality in important ways, leading to algorithms that work well in simulation and fail in reality. Closing the sim2real gap has gained significant interest, and various dynamics modeling approaches have been proposed (Figure 2 left).

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 [5, 33, 13, 29, 27, 7, 23, 31]. 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 [8, 32, 22], 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 [11, 6, 10, 21, 15, 14], light propagation [28, 16], and other phenomena [17, 24].

Residual physics models [34, 2, 19] augment physics engines with learned models to reduce the sim2real gap. Most of them introduce residual learning applied to the output of the physics engine, while we propose a more fine-grained approach, similar to Hwangbo et al. [19], where only at some parts in the simulator data-driven models are introduced. While in [19] the network for actuator dynamics is trained through supervised learning, our end-to-end differentiable model allows backpropagation of gradients from high-level states to any part of the computation graph, including neural network weights, so that these parameters can be optimized efficiently, for example from end-effector trajectories.

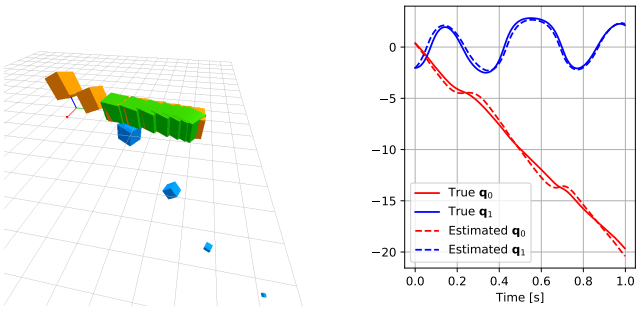


Fig. 1. *Left:* Trajectories from rigid body contact simulation of a cube thrown to the right. Starting with poor model parameters, the box falls through the ground (blue). After optimizing Equation 1, our simulation (orange) closely matches the target trajectory (green). *Right:* After system identification of a real double pendulum [4], the sim2real gap is strongly reduced.

II. Approach

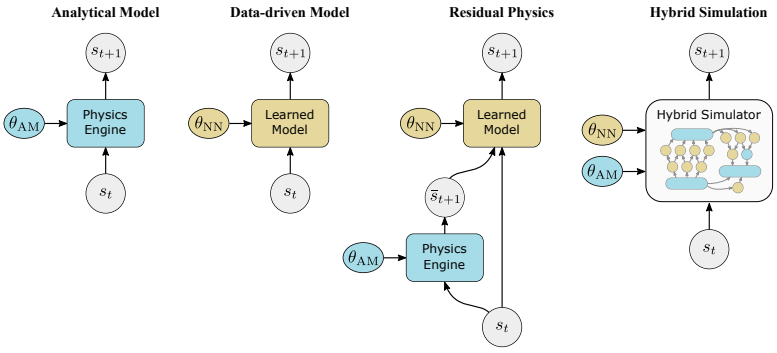
We propose a technique for hybrid simulation that leverages differentiable physics models and neural networks to allow for efficient system identification, design optimization, and gradient-based trajectory planning. By enabling any part of the simulation to be replaced or augmented by neural networks, we can learn unmodeled effects from data. Through template meta-programming, our open-source C++ implementation[1] allows any variable participating in the simulation to be augmented by neural networks that accept input connections from any other variable. In the simulation code, such *neural scalars* (Figure 2 right) are assigned a unique name, so that in a separate experiment code a "neural blueprint" is defined that declares the neural network architectures and sets the network weights. We compute gradients of the weights and analytical simulation parameters using the multi-dimensional dual number implementation from Ceres [1] and have support for many other automatic differentiation libraries.

III. System Identification

Given the state trajectory $\{s_t^*\}_{t=1}^T$ from the target system, we optimize the following loss for system identification:

$$\underset{\theta=[\theta_{AM},\theta_{NN}]}{\text{minimize}} \quad \mathcal{L} = \sum_t ||f_\theta(s_{t-1}) - s_t^*||^2 + R||\theta_{NN}||^2, \quad (1)$$

[1]https://github.com/google-research/tiny-differentiable-simulator



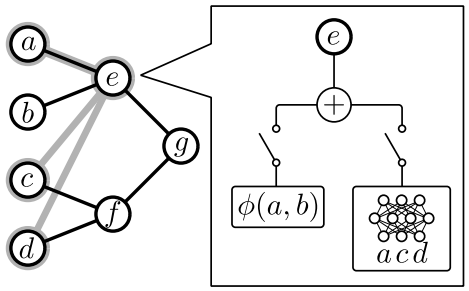


Fig. 2. *Left:* comparison of various model architectures (cf. Anurag et al. [2]). *Right:* augmentation of differentiable simulators with our proposed neural scalar type where variable e becomes a combination of an analytical model $\phi(\cdot, \cdot)$ with inputs a and b, and a neural network whose inputs are a, c, and d.

	Analytical	Data-driven	End2end ▽	Hybrid
Physics engine [8, 32, 22]	✓			
Residual physics [19, 2, 34]	✓	✓		✓
Learned physics [30, 5, 23, 20]		✓	✓	
Differentiable sim. [11, 17, 6, 10]	✓		✓	
Ours	✓	✓	✓	✓

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

where $f_\theta(\cdot)$ is the discrete dynamics function mapping from the previous simulation state s_{t-1} to the current state s_t which is implemented by our physics engine given the parameter vector θ that consists of the parameters θ_{AM} for the analytical model, plus the parameters θ_{NN} that correspond to the weights of the neural networks in the simulation. To ensure the residual dynamics learned by the neural networks are minimal, we regularize the network weights by factor R which penalizes large state contributions.

IV. Overcoming Local Optima

We solve the nonlinear least squares problem from Equation 1 using the Levenberg-Marquardt algorithm (LMA). Such gradient-based optimization method quickly finds local optima, but due to the highly nonconvex loss landscapes commonly encountered in system identification problems for nonlinear dynamics, the resulting parameter estimates often exhibit a poor fit to the real world data. To escape such poor local minima, we adapt a random search strategy, *parallel basin hopping* (PBH) [26], that, in our instantiation, runs multiple LMA solver and simulation instances in parallel while continuously randomizing the initial parameters from which the local solvers are restarted after convergence criteria, time limits, or maximum iteration counts are met.

V. Results

We present preliminary results for sim2sim transfer to match the hybrid dynamics model to richer analytical simulations. Additionally, we demonstrate our system identification approach on a real-world dataset.

In our first experiment, we transfer rigid-body contact dynamics simulated using a velocity-level contact model formulated as a linear complementarity problem [3]. Our hybrid simulator uses a point-based nonlinear-spring contact model where the normal force is solved analytically through the Hunt-Crossley model [18] and the friction force is learned by a neural network that receives the relative velocities, contact normal force and penetration depth as input. Before optimizing the analytical and neural model parameters, the trajectories of a cube thrown horizontally on the ground differ dramatically. After system identification using PBH applied on Equation 1 given trajectories of positions and velocities from the target system, the gap is significantly reduced (Figure 1 left).

In the next experiment, we apply our approach to a real-world dynamical system. Given joint position trajectories from the double-pendulum dataset provided by Asseman et al. [4], we optimize inertia, masses, and link lengths of our simulated model and achieve a minimal sim2real gap (Figure 1 right).

VI. Conclusion

We have demonstrated a simulation architecture that allows us to insert neural networks at any place in a differentiable physics engine to augment analytical models with the ability to learn dynamical effects from data. In our preliminary experiments, efficient gradient-based optimizers quickly converge to simulations that closely follow the observed trajectories from the target systems, while poor local minima are overcome through a random search strategy.

Future research is directed towards more automated ways to identify where such extra degrees of freedom are needed to close the sim2real gap given a few trajectories from the real system. Our loss function in Equation 1 regularizes the contributions of the neural networks to the overall system dynamics. Nonetheless, this approach does not prevent violating basic laws of physics, such as energy and momentum preservation. Hamiltonian [12] and (Deep) Lagrangian neural networks [25, 9] explicitly constrain the solution space to remain consistent with such principles but need to be further investigated in the context of residual models in hybrid simulators.

References

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

[2] 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.

[3] Mihai Anitescu and Florian A Potra. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics*, 14(3):231–247, 1997.

[4] Alexis Asseman, Tomasz Kornuta, and Ahmet Ozcan. Learning beyond simulated physics. In *Modeling and Decision-making in the Spatiotemporal Domain Workshop*, 2018. URL https://openreview.net/pdf?id=HylajWsRF7.

[5] 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.

[6] Justin Carpentier and Nicolas Mansard. Analytical derivatives of rigid body dynamics algorithms. In *Robotics: Science and Systems*, 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] Erwin Coumans et al. Bullet physics library. *Open source: bulletphysics.org*, 15(49):5, 2013.

[9] Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks, 2020.

[10] 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.

[11] 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.

[12] Sam Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. *arXiv preprint arXiv:1906.01563*, 2019.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] Kenneth H Hunt and Frank R Erskine Crossley. Coefficient of restitution interpreted as damping in vibroimpact. 1975.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

[25] Michael Lutter, Christian Ritter, and Jan Peters. Deep lagrangian networks: Using physics as model prior for deep learning. *arXiv preprint arXiv:1907.04490*, 2019.

[26] 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.

[27] 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.

[28] 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.

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

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

[32] 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.

[33] 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.

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