# End-to-End Differentiable Physics for Learning and Control

Filipe de Avila Belbute-Peres[1], Kevin Smith[2], Kelsey Allen[2], Josh Tenenbaum[2], J. Zico Kolter[13]
1. Carnegie Mellon University 2. Massachusetts Institute of Technology 3. Bosch Center for AI

## Introduction & Motivation

- Simulation-based models have strong physical knowledge and learn efficiently, but are inflexible; learned models are flexible, but require extensive (re)training and predictive precision decays fast
- We develop a differentiable physics engine, which has both precise knowledge of physics and can be embedded in an end-to-end learning system
- Previous similar work have either used numerical differentiation methods or relied solely on auto-differentiation. We propose finding the analytic gradients by differentiating the dynamics equations
- This system contributes to a recent trend of incorporating components with structured modules into end-to-end learning systems such as deep networks.

## The Engine

- Rigid body dynamics can be framed as an LCP:

$$\begin{bmatrix} 0 \\ 0 \\ a \\ \sigma \\ \zeta \end{bmatrix} - \begin{bmatrix} \mathcal{M} & -\mathcal{J}_e & -\mathcal{J}_c & -\mathcal{J}_f & 0 \\ \mathcal{J}_c & 0 & 0 & 0 & 0 \\ \mathcal{J}_f & 0 & 0 & 0 & E \\ 0 & 0 & \mu & -E^T & 0 \end{bmatrix} \begin{bmatrix} v_{t+h} \\ \lambda_e \\ \lambda_c \\ \lambda_f \\ \gamma \end{bmatrix} = \begin{bmatrix} \mathcal{M}v_t + hf \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\text{subject to: } \begin{bmatrix} a \\ \sigma \\ \zeta \end{bmatrix} \geq \mathbf{0}, \begin{bmatrix} \lambda_c \\ \lambda_f \\ \gamma \end{bmatrix} \geq \mathbf{0}, \begin{bmatrix} a \\ \sigma \\ \zeta \end{bmatrix}^T \begin{bmatrix} \lambda_c \\ \lambda_f \\ \gamma \end{bmatrix} = 0$$

- Solvable via primal-dual interior point method
- By differentiating the KKT conditions at a solution point we get the analytic gradients w.r.t to the inputs above (for some loss $\ell$):

$$\frac{\partial \ell}{\partial q} = -d_x \qquad \frac{\partial \ell}{\partial \mathcal{M}} = -\frac{1}{2}(d_x x^T + x d_x^T)$$
$$\frac{\partial \ell}{\partial m} = D(z^\star) d_z \qquad \frac{\partial \ell}{\partial G} = -D(z^\star)(d_z x^T + z d_x^T)$$
$$\frac{\partial \ell}{\partial A} = -d_y x^T - y d_x^T \qquad \frac{\partial \ell}{\partial F} = -D(z^\star) d_z z^T$$

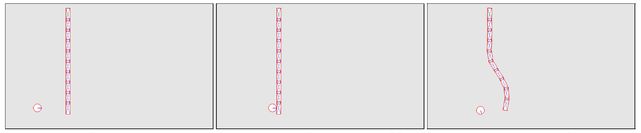
where:

$$x := -v_{t+dt} \quad q := -\mathcal{M}v_t - dtf_t \quad s := \begin{bmatrix} a \\ \sigma \\ \zeta \end{bmatrix}$$
$$y := \lambda_c \quad A := \mathcal{J}_c$$
$$z := \begin{bmatrix} \lambda_c \\ \lambda_f \\ \gamma \end{bmatrix} \quad G := \begin{bmatrix} \mathcal{J}_c & 0 \\ \mathcal{J}_f & 0 \\ 0 & 0 \end{bmatrix} \quad m := \begin{bmatrix} c \\ 0 \\ 0 \end{bmatrix} \quad F := \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & E \\ \mu & -E^T & 0 \end{bmatrix}$$

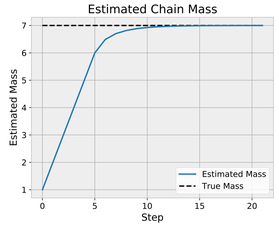
$$\begin{bmatrix} d_x \\ d_z \\ d_y \end{bmatrix} := \begin{bmatrix} \mathcal{M} & G^T & A^T \\ D(z^\star)G & D(Gx^\star + Fz^\star - m) + F & 0 \\ A & 0 & 0 \end{bmatrix}^{-T} \begin{bmatrix} (\frac{\partial \ell}{\partial x^\star})^T \\ 0 \\ 0 \end{bmatrix}$$

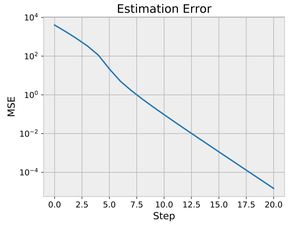
## Parameter learning

- A ball of known mass hits a chain. Positions of the objects are observed for 10s. Task is inferring the mass of the chain

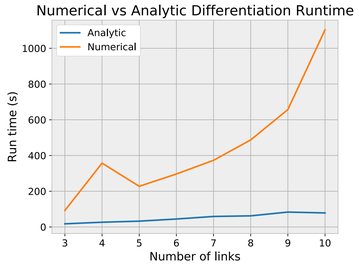


- Simulations are iteratively unrolled starting with an arbitrarily starting mass. The MSE between the observed positions and the simulated positions is observed, and then used to obtain its gradient w.r.t the estimated chain mass, which is optimized via gradient descent



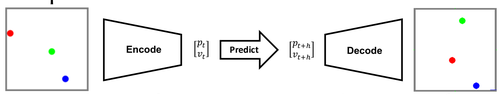




- The engine's analytic gradients are compared to numerical gradients (using finite-differences). Numerical gradients were more computationally expensive for a given number of parameters




## Visual prediction

- After observing 3 frames of a billiard ball-like scene, predict positions 10 frames into the future




- Autoencoder architecture. *Encoder* maps frames into physical predictions. *Engine* steps physics into the future. *Decoder* draws image from physics state
- Training is performed with only partially labeled data. For unlabeled examples, prediction uses the estimated parameters (notice the hats):

$$\hat{\phi}_t = encoder(x), \quad \hat{\phi}_{t+dt} = physics(\hat{\phi}_t), \quad \hat{y} = decoder(\hat{\phi}_{t+dt})$$

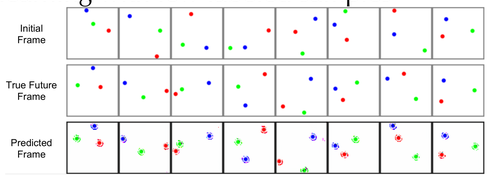
- When labels are available, prediction uses the true labels (notice the lack of hats):

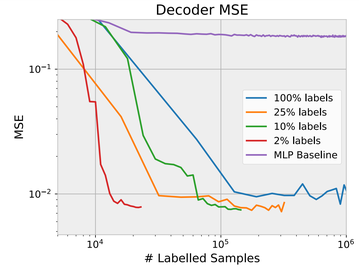
$$\hat{\phi}_t = encoder(x), \quad \hat{\phi}_{t+dt} = physics(\phi_t), \quad \hat{y} = decoder(\phi_{t+dt})$$

- Loss is composed of three terms when labels are available, or only the decoder loss when unlabeled

$$\mathcal{L} = \mathcal{L}_{enc} + \mathcal{L}_{phys} + \mathcal{L}_{dec},$$
$$\mathcal{L}_{enc} = \ell(\hat{\phi}_t, \phi_t), \quad \mathcal{L}_{phys} = \ell(\hat{\phi}_{t+dt}, \phi_{t+dt}), \quad \mathcal{L}_{dec} = \ell(\hat{y}, y)$$

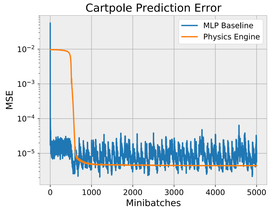
- Physics structure embedded into the model allows for learning with few labeled samples

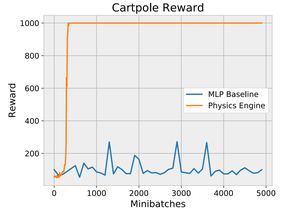


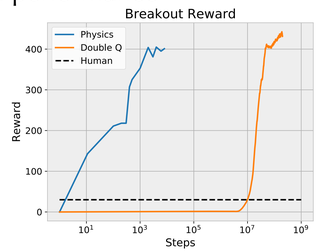

## Control

- Since the physics engine is differentiable, we use it in conjunction with iLQR for control in the *Cartpole* and the Atari *Breakout* tasks
- Simulation parameters (such as masses and sizes in Cartpole and paddle velocity in Breakout) are learned from unrolled simulations with random policy like in the parameter learning experiment
- Control performance is tested as parameters are learned. In the Cartpole environment high reward is achieved even before a perfect model is learned.




- The Breakout model with fitted parameters performs above human level and close to Double Q Network performance




## External resources

- Code available at: `https://github.com/locuslab/lcp-physics`