Physics Integration via Regression

Shayan Hoshyari, Chenxi Liu

April, 2018

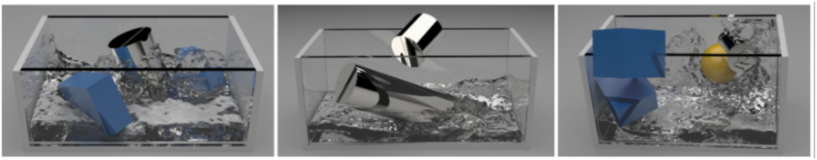
Motivation

Physics simulations can sometimes be cast as regression problems.

Motivation

Physics simulations can sometimes be cast as regression problems.

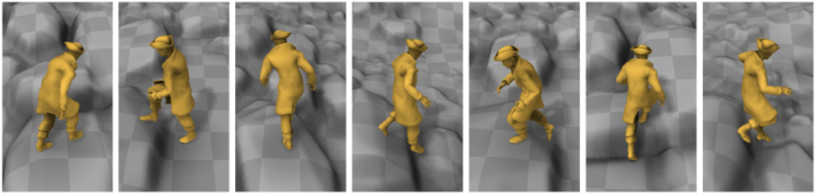
- Speed.



[Ladicky, et. al. SIGGRAPH Asia 2015]

Physics simulations can sometimes be cast as regression problems.

- Speed.
- Only data is available.



[Holden, et. al. SIGGRAPH 2017]

The problem

- Dynamical system, $\dot{\mathbf{s}} = \mathbf{f}(t, \mathbf{s}, \mathbf{u})$

  $\mathbf{s}$: state variables

  $\mathbf{u}$: control inputs

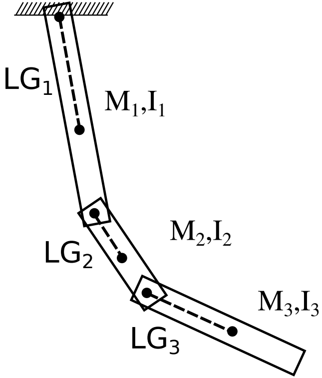
The problem

- Dynamical system, $\dot{\mathbf{s}} = \mathbf{f}(t, \mathbf{s}, \mathbf{u})$

  s: state variables

  u: control inputs

- Example, three link pendulum:



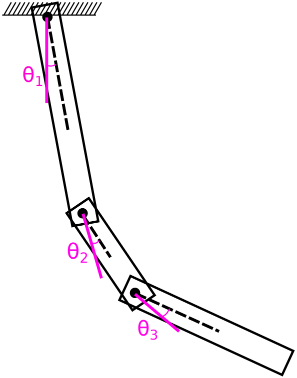
The problem

- Dynamical system, $\dot{\mathbf{s}} = \mathbf{f}(t, \mathbf{s}, \mathbf{u})$

  $\mathbf{s}$: state variables

  $\mathbf{u}$: control inputs

- Example, three link pendulum:



$$\mathbf{s} = [\theta_1, \theta_2, \theta_3, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3]$$

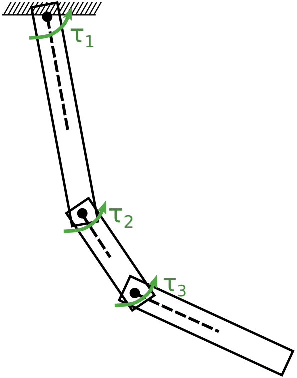
The problem

- Dynamical system, $\dot{\mathbf{s}} = \mathbf{f}(t, \mathbf{s}, \mathbf{u})$

  $\mathbf{s}$: state variables

  $\mathbf{u}$: control inputs

- Example, three link pendulum:



$$\mathbf{s} = [\theta_1, \theta_2, \theta_3, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3]$$

$$\mathbf{u} = [\tau_1, \tau_2, \tau_3]$$

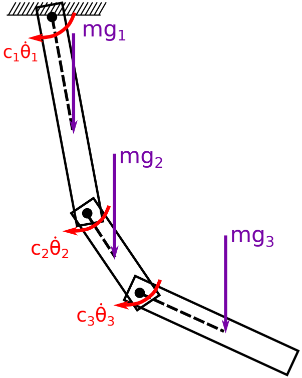
The problem

- Dynamical system, $\dot{\mathbf{s}} = \mathbf{f}(t, \mathbf{s}, \mathbf{u})$

  $\mathbf{s}$: state variables

  $\mathbf{u}$: control inputs

- Example, three link pendulum:



$\mathbf{s} = [\theta_1, \theta_2, \theta_3, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3]$

$\mathbf{u} = [\tau_1, \tau_2, \tau_3]$

Other forces included in $\mathbf{f}$

The problem – continued

Neuroanimator: Fast neural network emulation and control of physics-based models.

Grzeszczuk, Radek, Demetri Terzopoulos, and Geoffrey Hinton. SIGGRAPH 1998.

Given a pendulum with fixed properties (fixed $\mathbf{f}$),

The problem – continued

Neuroanimator: Fast neural network emulation and control of physics-based models.

Grzeszczuk, Radek, Demetri Terzopoulos, and Geoffrey Hinton. SIGGRAPH 1998.

Given a pendulum with fixed properties (fixed $\mathbf{f}$),

- Train a neural network to solve:

The problem – continued

Neuroanimator: Fast neural network emulation and control of physics-based models.

Grzeszczuk, Radek, Demetri Terzopoulos, and Geoffrey Hinton. SIGGRAPH 1998.

Given a pendulum with fixed properties (fixed $\mathbf{f}$),

- Train a neural network to solve:

  $\dot{\mathbf{s}} = \mathbf{f}(t, \mathbf{s}, \mathbf{u}), \mathbf{s} = \mathbf{s}_0, \mathbf{u} = \mathbf{u}(t),$

The problem – continued

Neuroanimator: Fast neural network emulation and control of physics-based models.

Grzeszczuk, Radek, Demetri Terzopoulos, and Geoffrey Hinton. SIGGRAPH 1998.

Given a pendulum with fixed properties (fixed $\mathbf{f}$),

- Train a neural network to solve:

  $\dot{\mathbf{s}} = \mathbf{f}(t, \mathbf{s}, \mathbf{u}), \mathbf{s} = \mathbf{s}_0, \mathbf{u} = \mathbf{u}(t),$

  i.e., the behaviour of the system.

The problem – continued

Neuroanimator: Fast neural network emulation and control of physics-based models.

Grzeszczuk, Radek, Demetri Terzopoulos, and Geoffrey Hinton. SIGGRAPH 1998.

Given a pendulum with fixed properties (fixed $\mathbf{f}$),

- Train a neural network to solve:

  $\dot{\mathbf{s}} = \mathbf{f}(t, \mathbf{s}, \mathbf{u}), \mathbf{s} = \mathbf{s}_0, \mathbf{u} = \mathbf{u}(t),$

  i.e., the behaviour of the system.

  which is "visually" acceptable.

The problem – continued

Neuroanimator: Fast neural network emulation and control of physics-based models.

Grzeszczuk, Radek, Demetri Terzopoulos, and Geoffrey Hinton. SIGGRAPH 1998.

Given a pendulum with fixed properties (fixed $\mathbf{f}$),

- Train a neural network to solve:

  $\dot{\mathbf{s}} = \mathbf{f}(t, \mathbf{s}, \mathbf{u}), \mathbf{s} = \mathbf{s}_0, \mathbf{u} = \mathbf{u}(t)$,

  i.e., the behaviour of the system.

  which is "visually" acceptable.
- Implement a control algorithm,

The problem – continued

Neuroanimator: Fast neural network emulation and control of physics-based models.

Grzeszczuk, Radek, Demetri Terzopoulos, and Geoffrey Hinton. SIGGRAPH 1998.

Given a pendulum with fixed properties (fixed $\mathbf{f}$),

- Train a neural network to solve:

  $\dot{\mathbf{s}} = \mathbf{f}(t, \mathbf{s}, \mathbf{u}), \mathbf{s} = \mathbf{s}_0, \mathbf{u} = \mathbf{u}(t)$,

  i.e., the behaviour of the system.

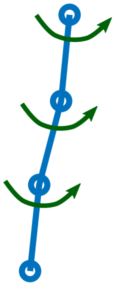
  which is "visually" acceptable.
- Implement a control algorithm,

  i.e., solve for $\mathbf{u}$ in the inverse problem:

  $\dot{\mathbf{s}} = \tilde{\mathbf{f}}(t, \mathbf{s}, \mathbf{u}), s = s(t)$

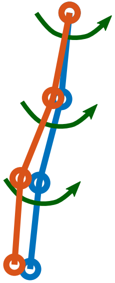
Learning problem

- Input: $\mathbf{y} = [\mathbf{s}_0, \mathbf{u}_0]$



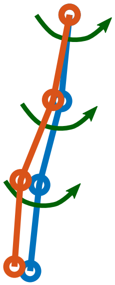
Learning problem

- Input: $\mathbf{y} = [\mathbf{s}_0, \mathbf{u}_0]$
- Regression Variable: $\mathbf{c} = [\mathbf{s}_{\Delta t} - \mathbf{s}_0]$, when $\mathbf{u}(t) = \mathbf{u}_0$



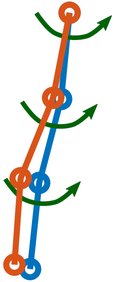
Learning problem

- Input: $\mathbf{y} = [\mathbf{s}_0, \mathbf{u}_0]$
- Regression Variable: $\mathbf{c} = [\mathbf{s}_{\triangle t} - \mathbf{s}_0]$, when $\mathbf{u}(t) = \mathbf{u}_0$
- $\mathbf{f}$ is found using the Lagrange equations.



Learning problem

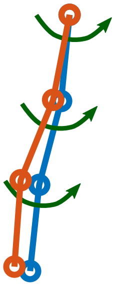
- Input: $\mathbf{y} = [\mathbf{s}_0, \mathbf{u}_0]$
- Regression Variable: $\mathbf{c} = [\mathbf{s}_{\triangle t} - \mathbf{s}_0]$, when $\mathbf{u}(t) = \mathbf{u}_0$
- $\mathbf{f}$ is found using the Lagrange equations.
- The ODE is solved for using `RK45`.



Learning problem

- Input: $\mathbf{y} = [\mathbf{s}_0, \mathbf{u}_0]$
- Regression Variable: $\mathbf{c} = [\mathbf{s}_{\triangle t} - \mathbf{s}_0]$, when $\mathbf{u}(t) = \mathbf{u}_0$
- $\mathbf{f}$ is found using the Lagrange equations.
- The ODE is solved for using `RK45`.
- Sampling:

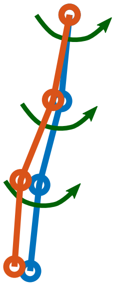
```
Y = [rand(n, 3)*r1 rand(n, 3)*r2 rand(n, 3)*r3]
```



- Input: $\mathbf{y} = [\mathbf{s}_0, \mathbf{u}_0]$
- Regression Variable: $\mathbf{c} = [\mathbf{s}_{\Delta t} - \mathbf{s}_0]$, when $\mathbf{u}(t) = \mathbf{u}_0$
- $\mathbf{f}$ is found using the Lagrange equations.
- The ODE is solved for using `RK45`.
- Sampling:

```
Y = [rand(n, 3)*r1 rand(n, 3)*r2 rand(n, 3)*r3]
```

- Paramters: Δt, r_1, r_2, r_3.



Neural network

- Single layer NN with hidden units: `[8*size(y)]`

Neural network

- Single layer NN with hidden units: `[8*size(y)]`
- Deep residual NN with hidden units: `[20, 20, 3*size(y)]`

Neural network

- Single layer NN with hidden units: `[8*size(y)]`
- Deep residual NN with hidden units: `[20, 20, 3*size(y)]`
- We did not perform an extensive search for the network architectures.

Neural network

- Single layer NN with hidden units: `[8*size(y)]`
- Deep residual NN with hidden units: `[20, 20, 3*size(y)]`
- We did not perform an extensive search for the network architectures.
- Optimization: SGD, non-linear CG, Newton with linesearch

Neural network

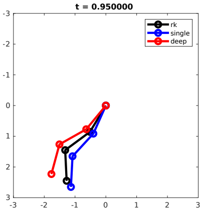
- Single layer NN with hidden units: `[8*size(y)]`
- Deep residual NN with hidden units: `[20, 20, 3*size(y)]`
- We did not perform an extensive search for the network architectures.
- Optimization: SGD, **non-linear CG**, Newton with linesearch

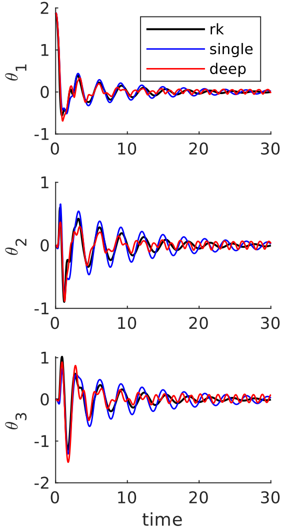
  `minimize.m` by Carl Edward Rasmussen.

Results – no **u**

```
n_links=3; M=[1,1,1]; L=[1,1,1]; LG=L; g=10; I=[0,0,0]; c=[1,1,1];
n=5000; r1=2pi; r2=2;
```

Arch.	Train error	Test error
deep	4.13%	4.75%
single layer	7.35%	7.72%

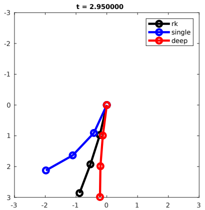


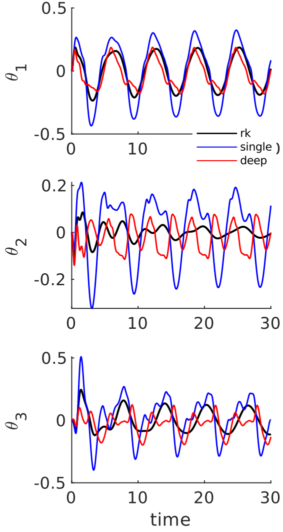


Results – with **u**

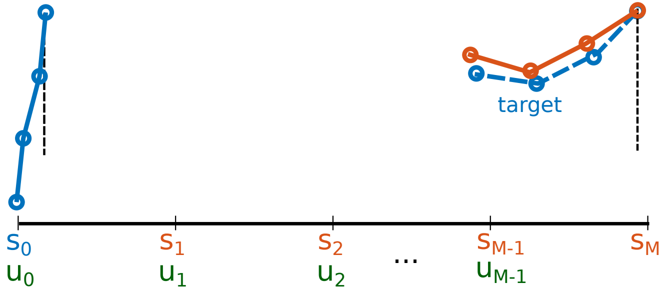
```
n_links=3; M=[1,1,1]; L=[1,1,1]; LG=L; g=10; I=[0,0,0]; c=[1,1,1];
n=5000; r1=2pi; r2=2; u=[5*cos(t),0,sin(t)]
```

Arch.	Train error	Test error
deep	7.23%	7.97%
single layer	10.57%	11.97%





Control problem



- s_0, target
- $\mathbf{U} = [\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{M-1}]$
- $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_M]$
- Control Problem:

  $\arg\min_{\mathbf{U}} E(\mathbf{U}; \text{target}, s_0) = \mu \|\mathbf{U}\|_2^2 + (s_M - \text{target})^2$

Gradient of control objective, $\nabla_U E$

Similar backpropagation problem as deep neural networks:

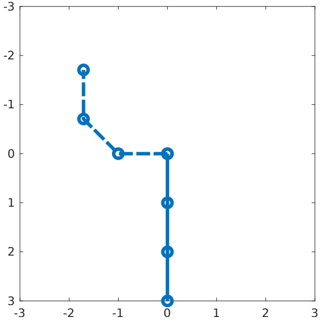
$$\begin{bmatrix} \boldsymbol{I} & \boldsymbol{I} + \partial_{\mathbf{s}_1} \mathbf{C}_2 & & & \\ & \ddots & & & \\ & & \boldsymbol{I} & \boldsymbol{I} + \partial_{\mathbf{s}_j} \mathbf{C}_{j+1} & \\ & & & \ddots & \\ & & & & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \partial_{\mathbf{s}_1} \mathbf{s}_M \\ \vdots \\ \partial_{\mathbf{s}_j} \mathbf{s}_M \\ \vdots \\ \partial_{\mathbf{s}_M} \mathbf{s}_M \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \vdots \\ \boldsymbol{I} \end{bmatrix}$$

$$\partial_{\mathbf{u}_j} \mathbf{s}_M = \partial_{\mathbf{s}_{j+1}} \mathbf{s}_M \left(\partial_{\mathbf{u}_j} \mathbf{C}_{j+1} + \boldsymbol{I} \right)$$

Control Results

Easier pendulum:

```
n_links = 3;
M = [1 1 1];
L = [1 1 1];
LG = [1, 1, 1];
g = 10;
I = [0,0,0];
c = [1, 1, 1];
```



More complicated physics:

```
n_links = 3;
M = [1 1.5 2];
L = [1 1.5 2];
LG = L ./ 2.;
g = 10;
I = 1/12 .* M .* L .* L;
c = [1, 1, 1];
```