

Interpolation of Molecular Dynamics Trajectories with Bi-Directional Neural Networks

Ludwig Winkler & Huziel Saucedo

November 16, 2020

Molecular Dynamics (MD)

- "Classical" dynamics of molecules are governed by Newton's equations of motion

Molecular Dynamics (MD)

- "Classical" dynamics of molecules are governed by Newton's equations of motion
- Position $r(t)$ and momentum $p(t)$ describe the system completely

Molecular Dynamics (MD)

- "Classical" dynamics of molecules are governed by Newton's equations of motion
- Position $r(t)$ and momentum $p(t)$ describe the system completely
- Dynamics f given by differential equation

$$\dot{x}(t) = \begin{bmatrix} \dot{r}(t) \\ \dot{p}(t) \end{bmatrix} = f \left(\begin{bmatrix} r(t) \\ p(t) \end{bmatrix}, t \right)$$

Molecular Dynamics (MD)

- "Classical" dynamics of molecules are governed by Newton's equations of motion
- Position $r(t)$ and momentum $p(t)$ describe the system completely
- Dynamics f given by differential equation

$$\dot{x}(t) = \begin{bmatrix} \dot{r}(t) \\ \dot{p}(t) \end{bmatrix} = f \left(\begin{bmatrix} r(t) \\ p(t) \end{bmatrix}, t \right)$$

- Solution is a trajectory in phase space

$$x(t) = \begin{bmatrix} r(t) \\ p(t) \end{bmatrix} = \int_{t_0}^t f \left(\begin{bmatrix} r(t) \\ p(t) \end{bmatrix}, t \right) dt$$

Molecular Dynamics (MD)

- High accuracy description of high dimensional N-bodies are expensive to compute

Molecular Dynamics (MD)

- High accuracy description of high dimensional N-bodies are expensive to compute
- MD requirements for a statistical meaningful result
 - $\Delta t = 0.2 \text{ fs}$ time step size for accurate integration of the equations of motion
 - $T \sim 1 \text{ ns}$: $10^6 - 10^7$ integration steps for expressive properties

Molecular Dynamics (MD)

- High accuracy description of high dimensional N-bodies are expensive to compute
- MD requirements for a statistical meaningful result
 - $\Delta t = 0.2 \text{ fs}$ time step size for accurate integration of the equations of motion
 - $T \sim 1 \text{ ns}$: $10^6 - 10^7$ integration steps for expressive properties
- Dynamics f are expensive to solve for long simulations

Molecular Dynamics (MD)

- High accuracy description of high dimensional N-bodies are expensive to compute
- MD requirements for a statistical meaningful result
 - $\Delta t = 0.2 \text{ fs}$ time step size for accurate integration of the equations of motion
 - $T \sim 1 \text{ ns}$: $10^6 - 10^7$ integration steps for expressive properties
- Dynamics f are expensive to solve for long simulations
- Single Thread DFT: 10s integration step requires 110 days

Molecular Dynamics (MD)

- High accuracy description of high dimensional N-bodies are expensive to compute
- MD requirements for a statistical meaningful result
 - $\Delta t = 0.2 \text{ fs}$ time step size for accurate integration of the equations of motion
 - $T \sim 1 \text{ ns}$: $10^6 - 10^7$ integration steps for expressive properties
- Dynamics f are expensive to solve for long simulations
- Single Thread DFT: 10s integration step requires 110 days
- Yet, essentially we are dealing with a time series prediction problem ...

Molecular Dynamics (MD)

- High accuracy description of high dimensional N-bodies are expensive to compute
- MD requirements for a statistical meaningful result
 - $\Delta t = 0.2 \text{ fs}$ time step size for accurate integration of the equations of motion
 - $T \sim 1 \text{ ns}$: $10^6 - 10^7$ integration steps for expressive properties
- Dynamics f are expensive to solve for long simulations
- Single Thread DFT: 10s integration step requires 110 days
- Yet, essentially we are dealing with a time series prediction problem ...

Can we learn the phase space dynamics with a ML algorithm?

Learning Dynamical Systems

- Learn dynamics f_θ with NN from true dynamics f

Learning Dynamical Systems

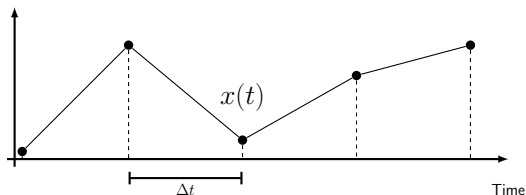
- Learn dynamics f_θ with NN from true dynamics f
- Integrate dynamics $\hat{\dot{x}}(t) = f_\theta$ to obtain learned solution $\hat{x}(t)$

Learning Dynamical Systems

- Learn dynamics f_θ with NN from true dynamics f
- Integrate dynamics $\hat{x}(t) = f_\theta$ to obtain learned solution $\hat{x}(t)$
- Optimize f_θ to predict the true solutions $x(t)$

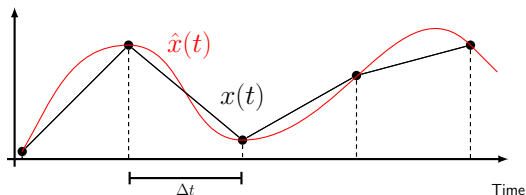
Learning Dynamical Systems

- Learn dynamics f_θ with NN from true dynamics f
- Integrate dynamics $\hat{\dot{x}}(t) = f_\theta$ to obtain learned solution $\hat{x}(t)$
- Optimize f_θ to predict the true solutions $x(t)$



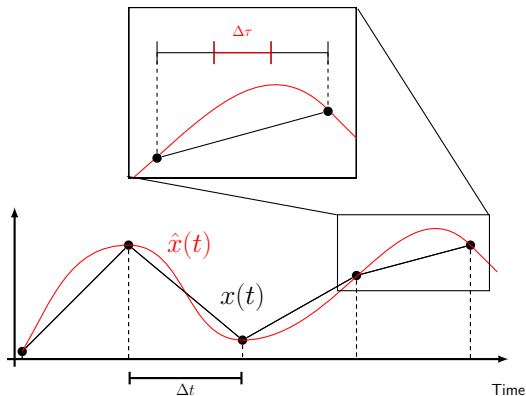
Learning Dynamical Systems

- Learn dynamics f_θ with NN from true dynamics f
- Integrate dynamics $\hat{\dot{x}}(t) = f_\theta$ to obtain learned solution $\hat{x}(t)$
- Optimize f_θ to predict the true solutions $x(t)$



Learning Dynamical Systems

- Learn dynamics f_θ with NN from true dynamics f
- Integrate dynamics $\hat{x}(t) = f_\theta$ to obtain learned solution $\hat{x}(t)$
- Optimize f_θ to predict the true solutions $x(t)$



Learning Dynamical Systems

Model Architectures

- ODENetwork

$$\dot{x}(t) = \begin{bmatrix} \dot{r}(t) \\ \dot{p}(t) \end{bmatrix} = f \left(\begin{bmatrix} r(t) \\ p(t) \end{bmatrix}, t \right)$$

Learning Dynamical Systems

Model Architectures

- ODENetwork

$$\dot{x}(t) = \begin{bmatrix} \dot{r}(t) \\ \dot{p}(t) \end{bmatrix} = f \left(\begin{bmatrix} r(t) \\ p(t) \end{bmatrix}, t \right)$$

- HamiltonianNetwork

$$\dot{x}(t) = \begin{bmatrix} \dot{r}(t) \\ \dot{p}(t) \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{H}(r(t), p(t), t)}{\partial p(t)} \\ -\frac{\partial \mathcal{H}(r(t), p(t), t)}{\partial r(t)} \end{bmatrix}$$

Learning Dynamical Systems

Model Architectures

- ODENetwork

$$\dot{x}(t) = \begin{bmatrix} \dot{r}(t) \\ \dot{p}(t) \end{bmatrix} = f \left(\begin{bmatrix} r(t) \\ p(t) \end{bmatrix}, t \right)$$

- HamiltonianNetwork

$$\dot{x}(t) = \begin{bmatrix} \dot{r}(t) \\ \dot{p}(t) \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{H}(r(t), p(t), t)}{\partial p(t)} \\ -\frac{\partial \mathcal{H}(r(t), p(t), t)}{\partial r(t)} \end{bmatrix}$$

- RNN and LSTM

$$\dot{x}(t) = \begin{bmatrix} \dot{r}(t) \\ \dot{p}(t) \end{bmatrix} = f_{\theta} \left(\begin{bmatrix} r(t_0) \\ p(t_0) \end{bmatrix}, \dots, \begin{bmatrix} r(t) \\ p(t) \end{bmatrix} \right), t$$

Learning Dynamical Systems with LSTM

- Best performing due to fewest assumptions and flexible parameterization

Learning Dynamical Systems with LSTM

- Best performing due to fewest assumptions and flexible parameterization
- Memory cell $c(t)$ to selectively read and write information

Learning Dynamical Systems with LSTM

- Best performing due to fewest assumptions and flexible parameterization
- Memory cell $c(t)$ to selectively read and write information
- Outputs $[\dot{r}(t), \dot{p}(t)]^T$ are integrated to obtain solution $\hat{x}(t)$

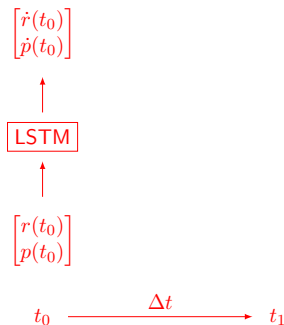
Learning Dynamical Systems with LSTM

- Best performing due to fewest assumptions and flexible parameterization
- Memory cell $c(t)$ to selectively read and write information
- Outputs $[\dot{r}(t), \dot{p}(t)]^T$ are integrated to obtain solution $\hat{x}(t)$



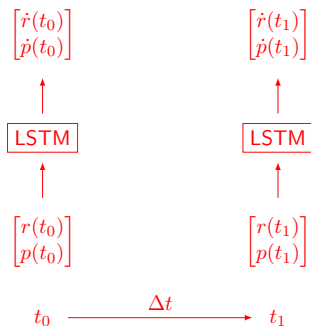
Learning Dynamical Systems with LSTM

- Best performing due to fewest assumptions and flexible parameterization
- Memory cell $c(t)$ to selectively read and write information
- Outputs $[\dot{r}(t), \dot{p}(t)]^T$ are integrated to obtain solution $\hat{x}(t)$



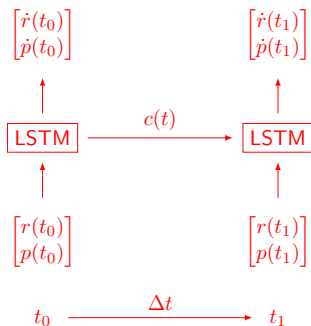
Learning Dynamical Systems with LSTM

- Best performing due to fewest assumptions and flexible parameterization
- Memory cell $c(t)$ to selectively read and write information
- Outputs $[\dot{r}(t), \dot{p}(t)]^T$ are integrated to obtain solution $\hat{x}(t)$



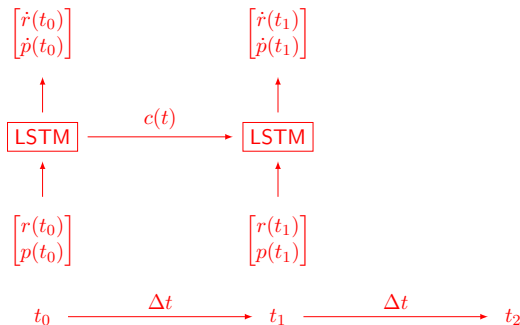
Learning Dynamical Systems with LSTM

- Best performing due to fewest assumptions and flexible parameterization
- Memory cell $c(t)$ to selectively read and write information
- Outputs $[\dot{r}(t), \dot{p}(t)]^T$ are integrated to obtain solution $\hat{x}(t)$



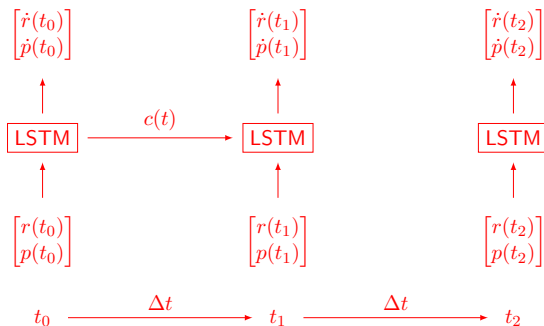
Learning Dynamical Systems with LSTM

- Best performing due to fewest assumptions and flexible parameterization
- Memory cell $c(t)$ to selectively read and write information
- Outputs $[\dot{r}(t), \dot{p}(t)]^T$ are integrated to obtain solution $\hat{x}(t)$



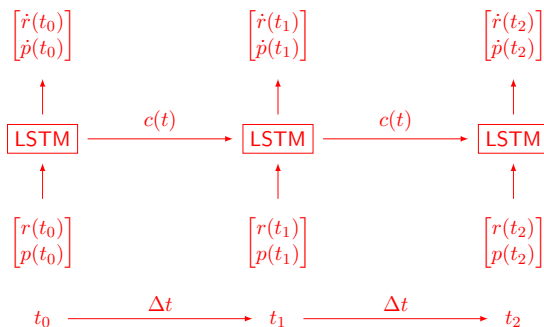
Learning Dynamical Systems with LSTM

- Best performing due to fewest assumptions and flexible parameterization
- Memory cell $c(t)$ to selectively read and write information
- Outputs $[\dot{r}(t), \dot{p}(t)]^T$ are integrated to obtain solution $\hat{x}(t)$



Learning Dynamical Systems with LSTM

- Best performing due to fewest assumptions and flexible parameterization
- Memory cell $c(t)$ to selectively read and write information
- Outputs $[\dot{r}(t), \dot{p}(t)]^T$ are integrated to obtain solution $\hat{x}(t)$



Bi-Directional Interpolation of Differential Equation

- Analytical simulation provides most accurate solution

Bi-Directional Interpolation of Differential Equation

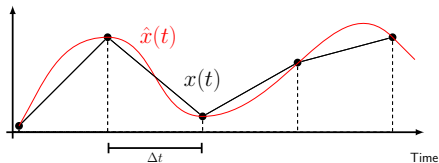
- Analytical simulation provides most accurate solution
- Combine speed of ML-MD with accuracy of analytical MD

Bi-Directional Interpolation of Differential Equation

- Analytical simulation provides most accurate solution
- Combine speed of ML-MD with accuracy of analytical MD
- Use coarse, analytical MD to provide initial and final conditions

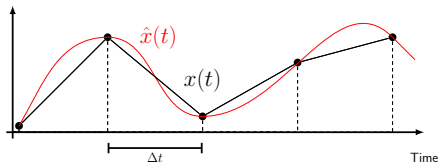
Bi-Directional Interpolation of Differential Equation

- Analytical simulation provides most accurate solution
- Combine speed of ML-MD with accuracy of analytical MD
- Use coarse, analytical MD to provide initial and final conditions



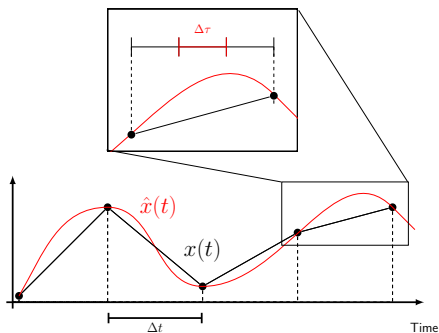
Bi-Directional Interpolation of Differential Equation

- Analytical simulation provides most accurate solution
- Combine speed of ML-MD with accuracy of analytical MD
- Use coarse, analytical MD to provide initial and final conditions



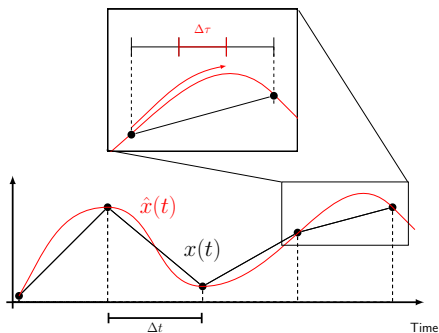
Bi-Directional Interpolation of Differential Equation

- Analytical simulation provides most accurate solution
- Combine speed of ML-MD with accuracy of analytical MD
- Use coarse, analytical MD to provide initial and final conditions



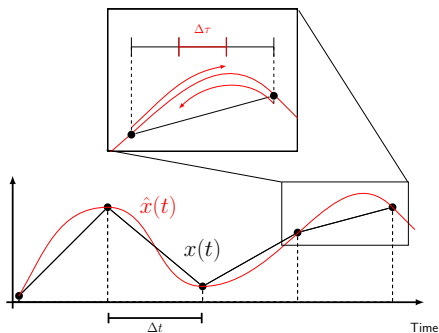
Bi-Directional Interpolation of Differential Equation

- Analytical simulation provides most accurate solution
- Combine speed of ML-MD with accuracy of analytical MD
- Use coarse, analytical MD to provide initial and final conditions



Bi-Directional Interpolation of Differential Equation

- Analytical simulation provides most accurate solution
- Combine speed of ML-MD with accuracy of analytical MD
- Use coarse, analytical MD to provide initial and final conditions



Bi-Directional Interpolation of Differential Equation

- Predict **forward solution** $\overrightarrow{\hat{x}(t)}$ and **backward solution** $\overleftarrow{\hat{x}(t)}$ with the **same** dynamics f_θ

Bi-Directional Interpolation of Differential Equation

- Predict **forward solution** $\overrightarrow{\hat{x}(t)}$ and **backward solution** $\overleftarrow{\hat{x}(t)}$ with the **same** dynamics f_θ
- Use adiabatic connection to interpolate $\overrightarrow{\hat{x}(t)}$ and $\overleftarrow{\hat{x}(t)}$ to $\hat{x}(t)$

Bi-Directional Interpolation of Differential Equation

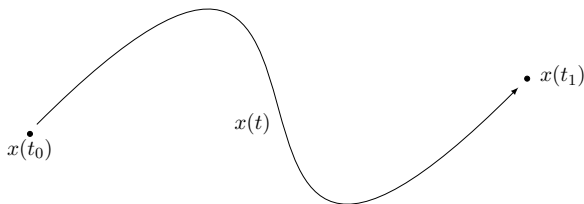
- Predict **forward solution** $\overrightarrow{\hat{x}(t)}$ and **backward solution** $\overleftarrow{\hat{x}(t)}$ with the **same** dynamics f_θ
- Use adiabatic connection to interpolate $\overrightarrow{\hat{x}(t)}$ and $\overleftarrow{\hat{x}(t)}$ to $\hat{x}(t)$

$$\hat{x}(t) = (1 - \lambda(t)) \overrightarrow{\hat{x}(t)} + \lambda(t) \overleftarrow{\hat{x}(t)} \quad (1)$$

Bi-Directional Interpolation of Differential Equation

- Predict **forward solution** $\overrightarrow{\hat{x}(t)}$ and **backward solution** $\overleftarrow{\hat{x}(t)}$ with the **same** dynamics f_θ
- Use adiabatic connection to interpolate $\overrightarrow{\hat{x}(t)}$ and $\overleftarrow{\hat{x}(t)}$ to $\hat{x}(t)$

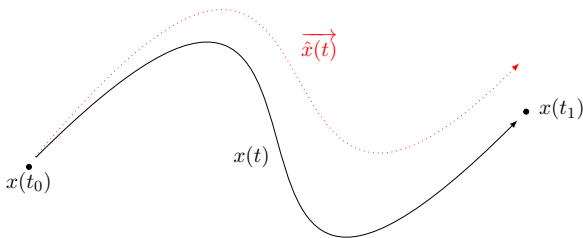
$$\hat{x}(t) = (1 - \lambda(t)) \overrightarrow{\hat{x}(t)} + \lambda(t) \overleftarrow{\hat{x}(t)} \quad (1)$$



Bi-Directional Interpolation of Differential Equation

- Predict **forward solution** $\overrightarrow{\hat{x}(t)}$ and **backward solution** $\overleftarrow{\hat{x}(t)}$ with the **same** dynamics f_θ
- Use adiabatic connection to interpolate $\overrightarrow{\hat{x}(t)}$ and $\overleftarrow{\hat{x}(t)}$ to $\hat{x}(t)$

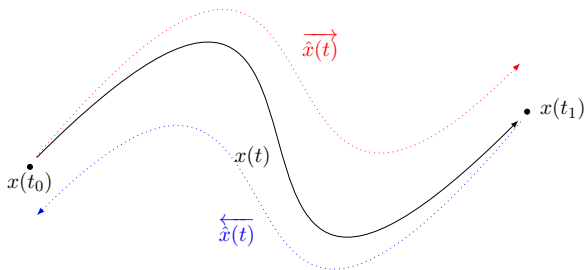
$$\hat{x}(t) = (1 - \lambda(t)) \overrightarrow{\hat{x}(t)} + \lambda(t) \overleftarrow{\hat{x}(t)} \quad (1)$$



Bi-Directional Interpolation of Differential Equation

- Predict **forward solution** $\overrightarrow{\hat{x}(t)}$ and **backward solution** $\overleftarrow{\hat{x}(t)}$ with the **same** dynamics f_θ
- Use adiabatic connection to interpolate $\overrightarrow{\hat{x}(t)}$ and $\overleftarrow{\hat{x}(t)}$ to $\hat{x}(t)$

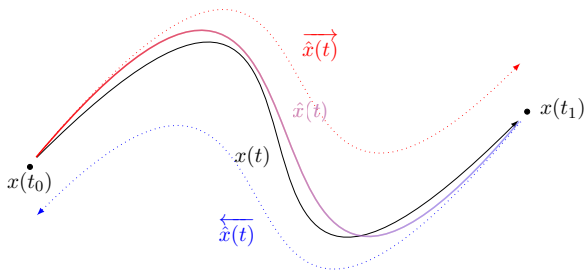
$$\hat{x}(t) = (1 - \lambda(t)) \overrightarrow{\hat{x}(t)} + \lambda(t) \overleftarrow{\hat{x}(t)} \quad (1)$$



Bi-Directional Interpolation of Differential Equation

- Predict **forward solution** $\overrightarrow{\hat{x}(t)}$ and **backward solution** $\overleftarrow{\hat{x}(t)}$ with the **same** dynamics f_θ
- Use adiabatic connection to interpolate $\overrightarrow{\hat{x}(t)}$ and $\overleftarrow{\hat{x}(t)}$ to $\hat{x}(t)$

$$\hat{x}(t) = (1 - \lambda(t)) \overrightarrow{\hat{x}(t)} + \lambda(t) \overleftarrow{\hat{x}(t)} \quad (1)$$

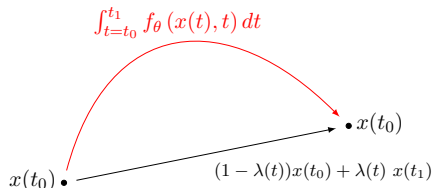


Bi-Directional Interpolation of Differential Equation

- For time-reversible solutions, we obtain

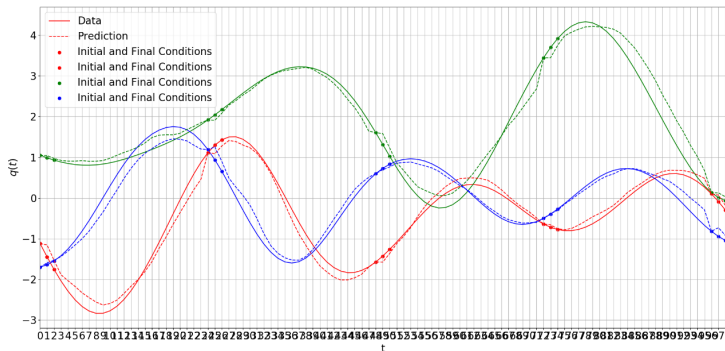
$$\begin{aligned}\hat{x}(t) &= (1 - \lambda(t)) \overrightarrow{\hat{x}(t)} + \lambda(t) \overleftarrow{\hat{x}(t)} \\ &= \underbrace{(1 - \lambda(t))x(t_0) + \lambda(t)x(t_1)}_{\text{low frequency components}} + \underbrace{\int_{t=t_0}^{t_1} f_{\theta}(x(t), t) dt}_{\text{high frequency components}}\end{aligned}$$

- Adiabatic connection frees the ML model to model high frequency signals



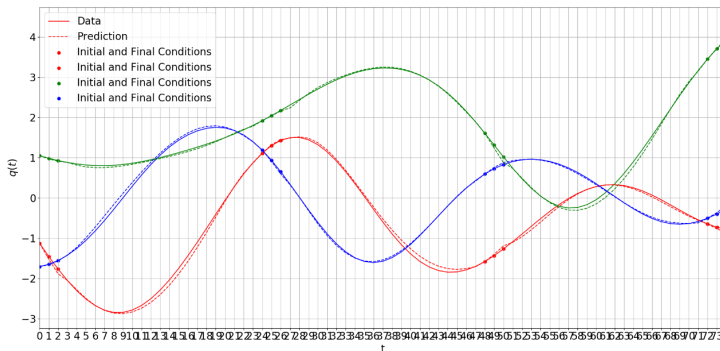
Bi-Directional Interpolation of Differential Equation

- Unidirectional LSTM architecture for Benzene MD trajectory interpolating over 20 time steps



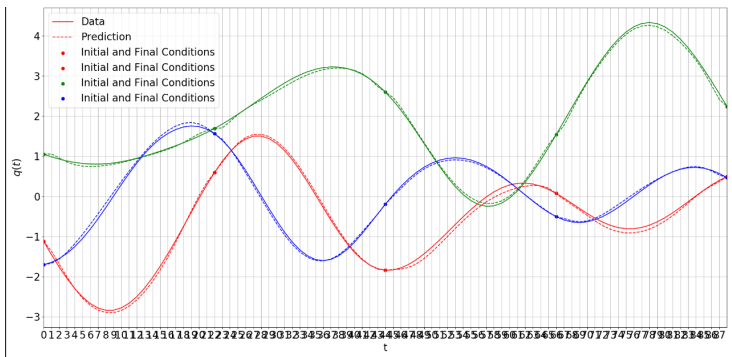
Bi-Directional Interpolation of Differential Equation

- Bidirectional LSTM architecture for Benzene MD trajectory interpolating over 20 time steps
- Final condition and additional bidirectional training smooth trajectories significantly



Bi-Directional Interpolation of Differential Equation

- Single initial and final condition already good for sufficient performance by bidirectional LSTM



Analysis of Interpolations

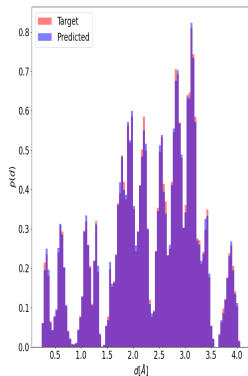


Figure:
Keto-Malonaldehyde
(100K)

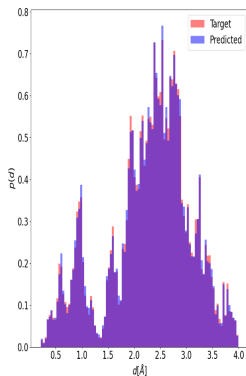


Figure:
Keto-Malonaldehyde
(300K)

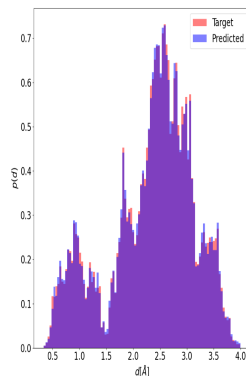


Figure:
Keto-Malonaldehyde
(500K)

Analysis of Interpolations

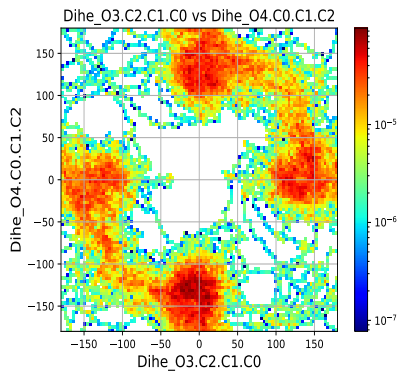


Figure: Ground Truth Free Energy
Keto-Malonaldehyde (300K)

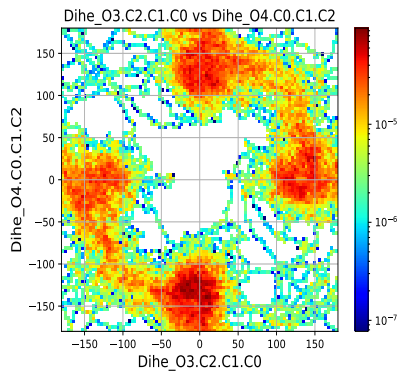


Figure: Predicted Free Energy
Keto-Malonaldehyde (300K)

Future Work

- Adaptively switch between simulation and ML prediction

Future Work

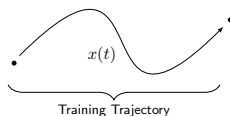
- Adaptively switch between simulation and ML prediction
- Regularly monitor ML prediction performance with parallel simulation

Future Work

- Adaptively switch between simulation and ML prediction
- Regularly monitor ML prediction performance with parallel simulation
- Leverage speed of ML prediction for smooth sub-trajectories

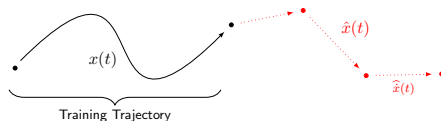
Future Work

- Adaptively switch between simulation and ML prediction
- Regularly monitor ML prediction performance with parallel simulation
- Leverage speed of ML prediction for smooth sub-trajectories



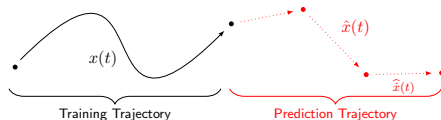
Future Work

- Adaptively switch between simulation and ML prediction
- Regularly monitor ML prediction performance with parallel simulation
- Leverage speed of ML prediction for smooth sub-trajectories



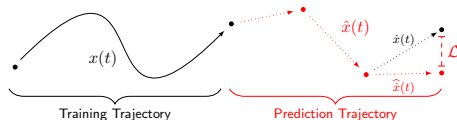
Future Work

- Adaptively switch between simulation and ML prediction
- Regularly monitor ML prediction performance with parallel simulation
- Leverage speed of ML prediction for smooth sub-trajectories



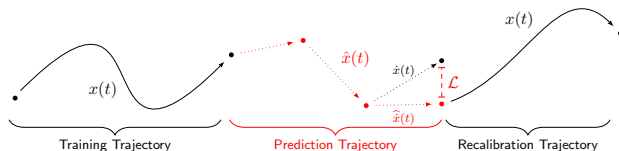
Future Work

- Adaptively switch between simulation and ML prediction
- Regularly monitor ML prediction performance with parallel simulation
- Leverage speed of ML prediction for smooth sub-trajectories



Future Work

- Adaptively switch between simulation and ML prediction
- Regularly monitor ML prediction performance with parallel simulation
- Leverage speed of ML prediction for smooth sub-trajectories



Future Work

- Adaptively switch between simulation and ML prediction
- Regularly monitor ML prediction performance with parallel simulation
- Leverage speed of ML prediction for smooth sub-trajectories

