Deep Lagrangian Networksを用いた剛体ロボットの逆動力学解析

○徐天涵　南裕樹　石川将人 （大阪大学）

Inverse Rigid-body Dynamics Analysis using Deep Lagrangian Networks

∗T. Xu, Y. Minami, M. Ishikawa (Osaka Univ.)

Abstract− In this paper, we use Deep Lagrangian Networks (DeLaN) that combine Lagrangian mechanics and deep learning to model the rigid-body dynamics. In addition, we implemente the model using the Autograd function of the deep learning library, which allows us to build more complex neural networks in the dynamics model. We also evaluated the activation function of the neural networks and selected the most appropriate activation function to improve the model performance. Finally, we conduct experiments with a planar double pendulum model, and the experimental results proved the effectiveness of our methods.

Key Words: System Identification, Dynamics Model, Physics-based Learning, Deep Learning

1　はじめに

力学問題を扱う際に，従来手法では力学の事前知識を利用し，数式などを使って力学モデルを表現する．しかし，現実問題には非線形摩擦などの定式化が難しい非線形項が含まれており，数式のみを使って詳細なモデルを構築することは容易でない．

この問題は，機械学習手法で非線形要素を近似することで回避できることがある．しかし，機械学習のみを利用した手法は力学の事前知識を一切使わずに学習データから物理モデルの特徴を学習しているため，モデルに物理法則のような一般性・汎用性が保証されない可能性がある．

つまり，従来手法のような力学の事前知識を利用したモデル同定では非線形項などの複雑な項を正確に表現できず，このときモデルは高バイアス (bias) になる．一方，機械学習手法のみを利用した方法では学習データに存在するランダムなノイズを学習してしまうことがあり，このときモデルは高バリアンス (variance) になる．

上記の背景から，最近では Physics-Based Deep Learning (PBDL) といった物理法則を考慮した深層学習手法が注目を集めている．本稿では，PBDL の一手法である，Deep Lagrangian Networks (DeLaN) [1] に注目し，それを剛体ロボットの逆動力学解析に応用する．力学的解析手法と機械学習手法を融合することにより，ロボットの自由度増加による運動方程式の複雑化や無視できない非線形摩擦の影響に関する問題を解決する．とくに，本稿では，文献 [1] に記載されているいくつかの問題点を改良し，さらに深層学習ライブラリの自動微分機能を利用してモデルを実装することにより，Deep Lagrangian Networks をベースとした深層力学フレームワークの汎用化，高性能化を図る．

2　剛体ロボットの逆動力学問題

2.1　モデル同定

本稿では，剛体ロボットの逆動力学モデルの同定を目標とする．

逆動力学とは，関節角などの情報から必要な関節トルクを求める問題である．関節角を $\mathbf{q}$，角速度と角加速度を $\dot{\mathbf{q}}, \ddot{\mathbf{q}}$，関節トルクを $\boldsymbol{\tau}$ とおくと，逆動力学問題は以下のように定式化できる．

$$\boldsymbol{\tau} = f(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \tag{1}$$

つまり，剛体ロボットのモデル同定は式 (1) の未知関数 f を求めることである．

2.2　運動モデル

多関節ロボットの運動方程式を考える．$L = T - V$ をラグランジアンとすると，ラグランジュの運動方程式は，

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{\mathbf{q}}} - \frac{\partial L}{\partial \mathbf{q}} = \boldsymbol{\tau} \tag{2}$$

と表される．運動エネルギー T は，

$$T = \frac{1}{2}\dot{\mathbf{q}}^{\mathsf{T}}\mathbf{H}(\mathbf{q})\dot{\mathbf{q}} \tag{3}$$

である．ここで，$\mathbf{H}$ は慣性行列と呼ばれる実対称正定値行列である．ポテンシャルエネルギー V は重力などの保存力によるポテンシャルであるため，

$$\frac{\partial V}{\partial \mathbf{q}} = \mathbf{g}(\mathbf{q}) \tag{4}$$

と表すことができる．この 2 つの関係式を式 (2) に代入すると，

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{H}}(\mathbf{q})\dot{\mathbf{q}} - \frac{1}{2}\left(\frac{\partial}{\partial \mathbf{q}}(\dot{\mathbf{q}}^{\mathsf{T}}\mathbf{H}(\mathbf{q})\dot{\mathbf{q}})\right)^{\mathsf{T}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} \tag{5}$$

のようになる．左辺の第 2, 3 項をまとめて $\mathbf{q}, \dot{\mathbf{q}}$ の関数で表すと運動方程式は，

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} \tag{6}$$

となる．

3 提案手法

3.1 ニューラルネットワークによるモデリング

本稿では，剛体ロボットの運動モデルをニューラルネットワークでモデリングすることを目標とする．提案手法のラグランジュ力学を考慮したニューラルネットワークモデルが Deep Lagrangian Networks (DeLaN) である．DeLaN の計算グラフを Fig. 1 に示す．

Deep Lagrangian Networks を π_ϕ とすると，運動モデルの同定は，

$$\hat{\phi} = \arg\min_{\phi} \|\pi_\phi(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) - \boldsymbol{\tau}\|_2 \tag{7}$$

を満たすパラメータ $\hat{\phi}$ を求めることになる．

式 (5) には 2 つの未知関数 $\mathbf{H}, \mathbf{g}$ があり，この二つの関数をニューラルネットワークでモデリングすることを考える．

まず，関数 $\mathbf{g}$ について考える．式 (5) より，$\mathbf{g}$ の計算には微分項がないため，全結合層からなるニューラルネットワーク $\hat{\mathbf{g}}(\mathbf{q}; \theta_g)$ を用いて表すことができる．ここで，θ_g は $\hat{\mathbf{g}}$ のパラメータを表す．

つぎに，関数 $\mathbf{H}$ について考える．$\mathbf{H}$ は対称正定値行列であるため，$\mathbf{H}$ の代わりに $\mathbf{L}\mathbf{L}^{\mathsf{T}} = \mathbf{H}$ のようにコレスキー分解した下三角行列 $\mathbf{L}$ を求める．また，正定値という制約条件があるため，$\mathbf{L}$ を正の対角成分とそれ以外の成分をベクトル化した $\mathbf{l}_d, \mathbf{l}_o$ の 2 つのベクトルに分けてニューラルネットワークを用いて回帰する．

$\mathbf{H}$ は $\mathbf{q}$ のみの関数であるため，状態 $\mathbf{q}$ の特徴抽出を行うニューラルネットワークを，

$$\mathbf{x}_h = \hat{h}(\mathbf{q}; \theta_h) \tag{8}$$

とする．ここで，$\hat{h}$ はニューラルネットワーク，θ_h は $\hat{h}$ のパラメータ，$\mathbf{x}_h$ は抽出された共通の特徴ベクトルである．この特徴ベクトルを用いて $\mathbf{l}_d$ と $\mathbf{l}_o$ の推測を行うニューラルネットワークをそれぞれ，

$$\mathbf{l}_d = \hat{\mathbf{l}}_d(\mathbf{x}_h; \theta_d) \tag{9}$$

$$\mathbf{l}_o = \hat{\mathbf{l}}_o(\mathbf{x}_h; \theta_o) \tag{10}$$

とする．ただし，対角成分を表す $\mathbf{l}_d$ の各成分が正となる必要があるため，出力の際に $x := \mathrm{ReLU}(x) + \mathrm{bias}$ の処理を施した．bias は小さな正の値である．これにより，下三角行列 $\mathbf{L}$ の推測ができ，$\mathbf{H}$ が求まる．

ここまでは普通のニューラルネットワークでの回帰であるが，式 (5) には $\mathbf{H}$ の時間微分と $\mathbf{q}$ に関する偏微分が含まれており，従来の手法ではこのような微分項を求めることができない．ここで，微分項の求め方について説明する．

時間微分

まずは時間微分について考える．$\mathbf{L}\mathbf{L}^{\mathsf{T}} = \mathbf{H}$ より，

$$\frac{\mathrm{d}\mathbf{H}}{\mathrm{d}t} = \mathbf{L}\frac{\mathrm{d}\mathbf{L}^{\mathsf{T}}}{\mathrm{d}t} + \frac{\mathrm{d}\mathbf{L}}{\mathrm{d}t}\mathbf{L}^{\mathsf{T}} \tag{11}$$

となる．したがって，$\dot{\mathbf{H}}$ を求めるには $\mathbf{L}$ の時間微分，つまり $\mathbf{l}_d$ と $\mathbf{l}_o$ の時間微分を求める必要がある．微分の連鎖律より，$\mathbf{l}_d$ の時間微分は，

$$\frac{\mathrm{d}\mathbf{l}_d}{\mathrm{d}t} = \frac{\mathrm{d}\mathbf{l}_d}{\mathrm{d}\mathbf{q}}\dot{\mathbf{q}} \tag{12}$$

となる．右辺の第 1 項は，微分の連鎖律によって求めることができる．また，$\dot{\mathbf{q}}$ はニューラルネットワークの入力変数ではないが，式 (1) の入力であるため既知である．以上より，$\dot{\mathbf{H}}$ を求めることができる．

偏微分

式 (5) の第 3 項には，$\mathbf{q}$ に関する偏微分が現れるが，この項を展開すると，

$$\frac{\partial}{\partial \mathbf{q}_i}(\dot{\mathbf{q}}^{\mathsf{T}}\mathbf{L}\mathbf{L}^{\mathsf{T}}\dot{\mathbf{q}}) = \dot{\mathbf{q}}^{\mathsf{T}}\frac{\partial \mathbf{L}}{\partial \mathbf{q}_i}\mathbf{L}^{\mathsf{T}}\dot{\mathbf{q}} + \dot{\mathbf{q}}^{\mathsf{T}}\mathbf{L}\frac{\partial \mathbf{L}^{\mathsf{T}}}{\partial \mathbf{q}_i}\dot{\mathbf{q}} \tag{13}$$

となる．$\mathbf{L}$ の偏微分も時間微分と同様に $\mathbf{l}_d$ と $\mathbf{l}_o$ の偏微分に分解でき，同じく微分の連鎖律を用いて求めることができる．

4 ニューラルネットワークモデルの改良

4.1 自動微分を用いた実装

文献 [1] では，最も単純な全結合層 (Fully-connected) ニューラルネットワークのみを使用しているため，式 (12), (13) に現れる運動方程式内の微分や偏微分の演算は微分の連鎖律により行列の積で計算することができる．このような単純なニューラルネットワークの性能は限られ，非線形項などの複雑な特徴を学習できない可能性がある．しかし，複雑なニューラルネットワークを実装しようとすると，微分や偏微分演算の実装はそれ以上に複雑になる．このような問題を解決

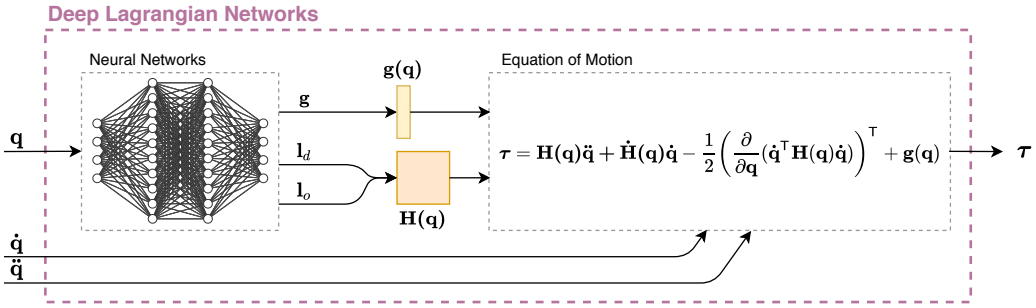

Fig. 1: Computational graph of Deep Lagrangian Networks

するため，本稿では深層学習ライブラリの自動微分機能を利用して微分演算を実装した．

両者は原理的には全く同じだが，自動微分を利用することによって複雑な微分計算を計算機に任せ，より複雑なニューラルネットワーク構造や dropout などのトリックの実装が可能になる．

4.2 活性化関数の選択

活性化関数はニューラルネットワークの重要な構成要素である．本稿では，活性化関数について調査を行い，実験を行った．

ニューラルネットワーク内の活性化関数

現在，ReLU 関数は計算が速い，勾配消失を解決できるなどの利点から多くのニューラルネットワークの活性化関数として使われている．しかし，ReLU への入力の一部がゼロ以下である場合，その微分もゼロになるためパラメータ更新ができず，Dying ReLU と呼ばれる問題が起きる場合がある．最近では，ニューラルネットワークにさまざまな正規化手法 (Batch Normalization など) を加えることによってこのような問題を回避することが試みられている．しかし，本稿で扱うモデルは物理法則に従うと仮定しているため，データのスケールを変更する正規化手法を用いることは適当でない．

そこで，活性化関数を見直すことにし，さまざまな活性化関数を用いて予備実験を行った．その結果，性能最良の LeakyReLU を採用してモデルの実装を行った．LeakyReLU 関数は以下の式で与えられる．

$$\text{LeakyReLU}(x) = \begin{cases} 0.01x & (x \leq 0) \\ x & (x > 0) \end{cases} \tag{14}$$

上式のように，LeakyReLU は入力がゼロ以下の場合でも小さな傾きを持ち，パラメータの更新が可能である．

慣性行列出力層の活性化関数

3.1 節では，ニューラルネットワークを使った慣性行列 $\mathbf{H}$ と重力項 $\mathbf{g}$ の計算方法を説明した．ここで，$\mathbf{H}$ は実対称正定値行列のため，コレスキー分解後の下三角行列 $\mathbf{L}$ の対角成分はすべて正である制約条件がある．

出力をすべて正の値にするため，$\mathbf{L}$ の対角成分 $\mathbf{l}_d$ を出力するニューラルネットワークの最後に $x := \text{ReLU}(x) + \text{bias}$ の処理を施した．しかし，ReLU 関数を使うと前述のような学習が収束しない問題以外にも，入力がゼロ以下の場合には慣性行列の対角成分の一部要素が全く同じ値になってしまい，運動モデルを正確に構築することができない．

そこで，さまざまな活性化関数で精度評価を行い，予測精度と収束性が最良の Softplus 関数を採用した．Softplus 関数は以下の式で与えられる．

$$\text{Softplus}(x) = \log(1 + e^x) \tag{15}$$

5 検証実験

5.1 平面 2 リンクマニピュレータの逆動力学解析

まず，解析解が分かっている平面 2 リンクマニピュレータで実験を行った．

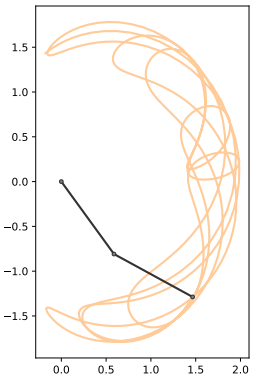

Fig. 2: Cosine trajectory of 2-DoF manipulator

簡単のため，長さ，質量などの物理量はすべて 1 として，シミュレーションで関節トルクの Ground Truth 値を求めた．2 リンクマニピュレータの角度変化は，以下の Cosine trajectory[1] に従うとする．

$$q^i(t) = q_0^i + A^i \cos\left(2\pi\omega^i\beta t + \sin(0.5\pi t)\right) \tag{16}$$

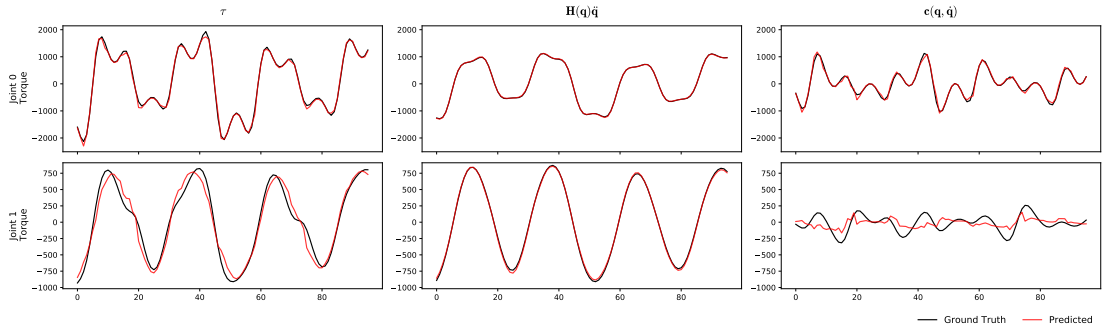

Fig. 3: Joint torque of 2-DoF manipulator on Cosine trajectory

Table 1: Parameters of Cosine trajectories

	Training set							Test set
ω^0	2	2	3	3	3	5	5	3
ω^1	5	7	5	7	11	11	13	7
β	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5

$i = 0, 1$ は関節の番号，q^i, q_0^i は i 番目の関節角度と初期角度，A^i, w^i はそれぞれ振幅定数と角速度定数，β は振動速度を調整するパラメータである．

Cosine trajectory の各パラメータを変化させてデータセットを作成し，学習データ : テストデータ $= 7 : 1$ の割合で実験を行った．今回の実験では $A^0 = 0.3\pi, A^1 = 0.5\pi$ とした．また，学習データとテストデータで使われた各パラメータを Table 1 に示す．

状態変数を $\mathbf{q} = (\theta_1, \theta_2)^\mathsf{T}$ とし，$\dot{\mathbf{q}}, \ddot{\mathbf{q}}$ を差分近似で求めた．モデルの入力を $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$，出力を関節トルク $\hat{\boldsymbol{\tau}} = \hat{\pi}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ とした．教師データとなる関節トルク $\boldsymbol{\tau}$ はラグランジュの運動方程式より解析的に求めた．誤差関数は Mean Squared Error(MSE), $\mathcal{L} = \|\boldsymbol{\tau} - \hat{\boldsymbol{\tau}}\|_2^2$ を採用した．学習後のモデルの予測結果を Fig. 3 に示す．

Fig. 3 より，逆動力学のモデリングができていることがわかる．2 リンクマニピュレータの実験では，慣性項 $\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}}$ とコリオリ力の項 $\mathbf{c}(\mathbf{q}, \dot{\mathbf{q}})$ の Ground Truth 値が計算できるため，モデルの予測値と比較した．

ここで，教師データとして与えているのはトルク $\boldsymbol{\tau}$ のみであり，$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}}$ と $\mathbf{c}(\mathbf{q}, \dot{\mathbf{q}})$ の情報は学習の際に使っていない．この 2 つの項の予測誤差も比較的に小さいため，モデルは単なる入出力のフィッティングだけでなく，潜在的な力学法則を学習できたといえる．

また，Table 1 からわかるように，学習データでの速度パラメータ β はすべて 1.0 に設定したが，テストデータでは 0.5 にした．これにより，異なる速度スケールに対するモデルのロバスト性を評価することができる．実際にパラメータ β を $0.5 \sim 2.0$ の範囲内で変化させた結果，すべて Fig. 3 のような結果となった．

5.2 考察

文献 [1] のように活性化関数として ReLU を使った場合，モデルの初期パラメータによって学習が収束しない場合があった．4 節で説明したような改良を施した結果，モデルの学習が進み，最終的には小さな誤差に収束する．

また，Fig. 3 より，Joint 1 のコリオリ力の項の推定誤差が比較的に大きい．これは，誤差関数は MSE を使用しているため，絶対値が大きい項の誤差を減らすためのパラメータの更新が優先的に行われ，絶対値が小さい Joint 1 のコリオリ力の項に誤差が残ってしまうと考えられる．

6 おわりに

本稿では，物理法則を融合した機械学習モデルの一つである Deep Lagrangian Networks を用いて運動モデルの構築を行った．

今後の課題として，自動微分機能を活用したより複雑なニューラルネットワークの実装，物理法則に従う前提での入出力データの正規化手法の検討，さらに高自由度な運動モデルを利用した実験などがあげられる．

参考文献

[1] Michael Lutter, Christian Ritter, and Jan Peters: Deep lagrangian networks: Using physics as model prior for deep learning; *International Conference on Learning Representations* (2019)

[2] Duy Nguyen-Tuong, Jan Peters, Matthias Seeger, and Bernhard Scholkopf: Learning inverse dynamics: A comparison; *The European Symposium on Artificial Neural Networks*, pp. 13–18 (2008)

[3] Jared Willard, Xiaowei Jia, Shaoming Xu, Michael S. Steinbach, and Vireshwar Kumar: Integrating physics-based modeling with machine learning: A survey; *ArXiv*, abs/2003.04919 (2020)