Model–Based Policies in Continuous Time, States and Actions: Surrogate Models and Gradient Estimation

Stefano Massaroli^{1,2}, Michael Poli^{3}, Ren Komatsu^{1}, Alessandro Moro^{1}, Atsushi Yamashita^{1}, Hajime Asama^{1}

***Abstract*— We discuss a learning–based treatment of optimal decision making in continuous time, states and actions, relying on direct optimization of policy networks. Specifically, we consider the problem of controlling physical systems with unknown dynamics and propose to learn a *surrogate* differential equation model alongside the policy through observations of the state. We show how model errors propagate to the Lagrangian multiplier of the policy optimization problem.**

I. Learning Policies in Continuous Time, States and Actions

We consider the following class of deterministic continuous–time dynamical systems:

$$\dot{s}_t = f(s_t, a_t), \quad t \in \mathbb{T} \tag{1}$$

with *state* $s_t \in \mathbb{S} \subset \mathbb{R}^n$ and *action* $a_t \in \mathbb{A} \subset \mathbb{R}^m$. The vector field $f : \mathbb{S} \times \mathbb{A} \to \mathbb{R}^n$ is assumed to be smooth enough so that solutions of (1) are forward complete on the compact time domain $\mathbb{T} := [0, T]$ for any initial condition $s_0 \in \mathbb{S}$. The initial conditions are assumed to be distributed according to $\rho_0(s)$ with support in $\mathbb{S}$.

In its simplest instance, optimal decision making in continuous time, and state–action spaces can be reduced to finding an *optimal policy* $\pi = \{a_t\}_{t\in\mathbb{T}}$ via the following nonlinear program:

$$\begin{aligned} \inf_{\pi} \quad & \mathbb{E}_{s_0}\left[J(s_0, \pi)\right] \\ \text{subject to} \quad & \dot{s}_t = f(s_t, a_t) \\ & a_t \in \pi \end{aligned} \tag{2}$$

where $J(s_0, \pi)$ is a smooth objective function measuring the performance of the policy on a given task. With $S_t = s_t(s_0, \pi)$ the trajectory of the system (i.e. the solution of (1)) at time $t \in \mathbb{T}$ given initial state s_0 and policy π, the objective J is usually designed to comprise a *running* (integral) term and a *terminal* term, e.g.

$$J(s_0, \pi) = \mathbb{E}_t\left[r_t(S_t, a_t)\right] + R(S_T, a_T), \tag{3}$$

In this manuscript we consider the following class of deterministic neural network policies [1].

Assumption (*Deterministic Policy*): Actions are parameterized by a neural network $\mu_\theta : \mathbb{S} \to \mathbb{A}$ with parameters θ, realizing the *state–feedback* map $s_t \mapsto a_t$, i.e. $a_t = \mu_\theta(s_t)$. The selection problem of a policy $\pi = \{\mu_\theta(s_t)\}_t$ corresponds to the choice of network parameters $\theta \in \mathbb{R}^p$.

If the dynamics (1) are known, the control problem is classified as *model–based*. Nonlinearities within the controlled vector field $f(s_t, a_t)$ and non–convexity of the desiderata $J(\cdot, \pi)$ render, in most cases, the control problem (2) intractable to be solved analytically. Numerical optimal policies can still be found through e.g. approximate *dynamic programming* [2], [3], [4] or relying on *Pontryiagin maximum principle* [5], [6]. Alternatively, *receding horizon* [7], [8], [9] and *shooting methods* [10], [11] can be used to directly optimized the parameters θ of the policy network.

A. Direct optimal policies via gradient descent

In this work, we iteratively refine an optimal policy by stochastic gradient descent on the policy parameters θ. As the policy is uniquely determined by the value of θ, we write π_θ. One iteration of the training procedure works as follows: an initial condition (or a batch of them) is sampled from the distribution ρ_0. Then, the initial value problem (IVP) is (numerically) integrated to *unroll* the episode. After the loss $J(s_0, \pi)$ is computed, the policy parameters are updated according to a chosen gradient–based optimizer.

$s_0 \sim \rho_0(s)$	1. sample initial conditions
$s_0, \theta \mapsto \{(s_t, a_t)\}_{t\in\mathbb{T}}$	2. unroll episode
$\{(s_t, a_t)\}_{t\in\mathbb{T}} \mapsto J(\pi_\theta)$	3. compute loss function
$\theta \leftarrow \theta - \eta\nabla_\theta J(\pi_\theta)$	4. update policy parameters

The implementation of such training algorithm necessitates of two distinct classes of numerical routines: a differential equations solver (to *unroll* the episode) and a differentiation algorithm to compute the loss gradient w.r.t. the policy parameters. In the typical model–free setting we think the environment as an *oracle* providing complete state trajectories given the initial state s_0 and the policy π. Additionally, here we can use the model to differentiate exactly (up to numerical integration errors) through the solution of (1) via e.g. the adjoint method [5], [12] or simply differentiating through the discrete steps of the forward numerical

*This work is supported by JST [Moonshot Research and Development], Grant Number [JPMJMS2032].

^{1}The University of Tokyo.
^{2}Mila – Quebec. stefano.massaroli@mila.quebec
^{3}Stanford University.

integration (*discrete adjoint*). In other words, we have an oracle for the gradients of policy parameters. Gradient based optimization of continuous–time neural policies is enabled by e.g. differentiable simulation environments [13], [14], [15], [16], [17] and numerical routines for differential equations integrated in popular deep learning software ecosystems such as `PyTorch` [18] or `JAX`. See, for example, `torchdyn` [19], `torchdiffeq` [20] or `diffrax` [21].

B. Stabilization as Maximum likelihood

Consider the *mean squared error* loss:

$$J(s_0, \pi_\theta) = \frac{1}{2}\mathbb{E}_t\left[\|s_T(s_0, \pi_\theta) - s^*\|^2\right], \quad s^* \in \mathbb{S} \tag{4}$$

penalizing the distance between final state s_T and the target s^*. Achieving stabilization of s^* by minimizing (4) corresponds (up to a normalizing constant) to a maximum likelihood problem $\inf_\theta -\log \mathcal{N}(S_T|s^*, \mathbb{I})$. The Fokker–Planck equation describes the evolution of the probability density function of a continuous–time stochastic process. Particularly, the probability density $\rho_t(s)$ of state s_t in (1) evolves according to

$$\partial_t \rho_t(s) = \boldsymbol{\nabla} \cdot (f(s, a_t)\rho_t(s))$$

It is thus possible to train deterministic policies that *transport* distributions $\rho_0(s)$ of initial states into target distribution $\rho_T^*(s)$ in a time T via the policy π_θ.

If $\rho_0(s)$ has tractable likelihood (or can be freely assigned) and a sampler is available for $\rho^*(s)$, policies can be efficiently trained in a *continuous normalizing flow* [22] fashion integrating the system backward in time and training on the exact log–likelihood

$$\log \rho^*(s_T) = \log \rho_0(s_0) - \int_0^T \boldsymbol{\nabla} \cdot f(s_t, a_t)\mathrm{d}t.$$

computed in expectation over trajectories of the system.

II. Surrogate ODE Models

When the system dynamics $f(s_t, a_t)$ are not known, it is no longer possible to exactly back–propagate through trajectories, loosing the *oracle* for policy gradients. A simple gradient estimator can be learned by training a neural approximator $\gamma_\omega : \mathbb{S} \times \mathbb{A} \to \mathbb{R}^n$ with parameters ω such that the forward simulation of the system

$$\dot{\hat{s}}_t = \gamma_\omega(\hat{s}_t, \hat{a}_t), \quad t \in \mathbb{T} \tag{5}$$

matches the evolution of the true state $\{s_t\}_{t\in\mathbb{T}}$. Then, the *surrogate model* naturally induces the gradient estimator $\nabla_\theta \hat{J}(s_0, \pi_\theta)$ for the policy gradients $\nabla_\theta J(s_0, \pi_\theta)$ where $\hat{J}(s_0, \pi_\theta)$ denotes the control objective computed with the estimated trajectories. In §III we measure the quality of such estimator highlighting some limitations of naive surrogate models in long time horizons.

The surrogate ODE model (5) belongs to the class of *neural differential equations* [20], [23], [24]. Neural DEs can be parameterized with different design flavors, e.g. VAE–like [25], using *rough paths* [21] or *state–space* models [26], [27].

Training can be achieved, for example, by maximizing the likelihood of some chosen prior $p(s_t)$ over trajectories[1] of the *simulated* system. A common choice for continuous–time learning models is $p(s_t) = \mathcal{N}(s_t, \sigma_t\mathbb{I})$ where σ_t is often taken increasing in time to account for epistemic uncertainty of predictions [25]. This corresponds to minimizing the log–likelihood

$$-\mathbb{E}_{s_0}\mathbb{E}_t\left[\log \mathcal{N}(\hat{s}_t(\pi_\theta)|s_t(\pi_\theta), \sigma_t\mathbb{I})\right]$$

Trajectories collected *off*–line with any policy can be used to pre–train the surrogate model (see e.g. [28], [29]). Greedy trajectories[2] observed during policy optimization steps are instead available to fine–tune the learned dynamics alongside with the policy. In the latter setting, an additional reconstruction metric is the discrepancy between the control objective computed on nominal and estimated trajectories, i.e.

$$\mathbb{E}_{s_0}[J(s_0, \pi_\theta) - \hat{J}(s_0, \pi_\theta)]$$

As the control objective is computed with the current candidate optimal policy, reduction of this discrepancy improves gradient estimates on the greedy paths.

> If the policy and the surrogate model parameters are optimized simultaneously (*on*–line setting), the policy induces the distribution of trajectories (with instantaneous probability $\rho_t(s)$) used to train γ_ω. Conversely, the surrogate model induces the gradient estimator.

A. Injecting inductive biases in the surrogate model

In practical control problems of physical systems we can inject domain knowledge on the underlying dynamics into the surrogate model. Most cases fall in the following two scenarios, listed by difficulty:

$i.$ The model is partially known, e.g.

$$\dot{s}_t = f(s_t, a_t) + g(s_t, a_t)$$

with a known term f and a *small* unknown term g.

$ii.$ The *physics* first principles of the process are known but no actual computational model is available.

In the first scenario, the surrogate model has to approximate only the discrepancy term $g(s_t, a_t)$. Here simple parameterizations are often sufficient to track the policy gradient, e.g. *Gaussian Processes* [30].

In the second scenario the surrogate model usually consists of a neural architecture *wrapped* with domain–specific first principles equations. Particularly appealing to the roboticist reader are *Lagrangian* and *Hamiltonian* variants [31], [32] of Neural ODEs.

It is worth to be noted that domain knowledge and inductive biases can also be injected in the policy network design, e.g. enforcing stability or passivity of the closed–loop system [33]).

[1]Collected either *on* and *off*–line.
[2]Computed with the current estimate of the optimal policy.

Surrogate ODE models trained for decision making are the continuous counterparts of notable model–based reinforcement learning algorithms such as PILCO [34], [35] *world models* [36], *decision transformers* [37], [38] or *generalist agents* [39].

III. Gradient Error Propagation Analysis

We can now proceed to analyze the performance of the gradient estimator. Specifically, the following will introduce the a bound on the estimation error of the Lagrange multiplier λ_t of the optimization problem[3] which carries most of the model uncertainty from the forward integration to the policy gradients. As the policy is a pure state feedback we write $\bar{f}_\theta(s_t) = f(s_t, \mu_\theta(s_t))$ and $\bar{\gamma}_{\omega,\theta} = \gamma_\omega(s_t, \mu_\theta(s_t))$. The following assumptions are considered:

i. The surrogate dynamics γ_ω are such that $\gamma_\omega(s_t, a_t) = f(s_t, a_t) + \varepsilon(s_t, a_t)$ with ε measurable on $\mathbb{T}$, $\exists \delta_f : \forall t \in \mathbb{T}\ 0 \leq \int_0^t \|\varepsilon(s_\tau, a_\tau)\| \mathrm{d}\tau \leq \delta_f$.

ii. The dynamics f, the surrogate dynamics γ, the residual ε and all their partial derivatives are Lipsichitz continuous w.r.t. all their arguments[4].

iii. The objective only comprises a terminal termm i.e. $J(s_0, \pi_\theta) = R(S_T)$.

Error Analysis: Tracing the model error propagation from the forward process to the Lagrange multiplier, we need to overcome a series of integral bounds. We leverage of smoothness assumptions of all functions involved to linearize such bounds and apply Gronwall's lemma.

Lemma 3.1 (Gronwall's Lemma[5] [41]): Let $\{x_t\}_t : \mathbb{T} \to \mathbb{R}$ and let $A \in \mathbb{R}$ and b_t a non–decreasing function. If $x_t \leq b_t + \int_0^t A x_r \,\mathrm{d}\tau$ then $x_t \leq e^{At} b_t$.

A. Model error bound $\mapsto$ state error bound

Let $\tilde{s}_t = s_t - \hat{s}_t$ be the state prediction error at time t. The error is then governed by the following dynamics

$$\tilde{s}_t = \int_0^t \left[\bar{f}_\theta(s_\tau) - \bar{\gamma}_{\omega,\theta}(\hat{s}_\tau)\right] \mathrm{d}\tau.$$

We want to find an upper bound for $\|\tilde{s}_t\|$. It holds

$$\begin{aligned}
\|\tilde{s}_t\| &= \left\| \int_0^t \left[\bar{f}_\theta(s_\tau) - \bar{\gamma}_{\omega,\theta}(\hat{s}_\tau)\right] \mathrm{d}\tau \right\| \\
&\leq \int_0^t \left\|\bar{f}_\theta(s_\tau) - \bar{f}_\theta(\hat{s}_\tau)\right\| \mathrm{d}\tau + \int_0^t \left\|\bar{\varepsilon}_\theta(\hat{s}_\tau)\right\| \mathrm{d}\tau \\
&\leq \int_0^t m_f \underbrace{\|s_\tau - \hat{s}_\tau\|}_{\|\tilde{s}_\tau\|} \mathrm{d}\tau + \delta_f \\
&\leq \delta_f e^{m_f t} \qquad \text{(Gronwall's Lemma).}
\end{aligned}$$

[3]see [12] for a detailed derivation of the adjoint equation driving the Lagrange multiplier

[4]For compactness, we denote all the Lipsichitz constants with m (e.g. m_f, m_γ, etc.), without explicitly declaring them.

B. State error bound $\mapsto$ action error bound

Similarly, we can bound the error we have on each action computed with the surrogate state $\tilde{a}_t = a_t - \hat{a}_t$

$$\|\tilde{a}_t\| = \|\mu_\theta(s_t) - \mu_\theta(\hat{s}_t)\| \leq m_\mu \|s_t - \hat{s}_t\| \leq m_\mu \delta_f e^{m_f t}.$$

C. Error propagation to Lagrange multiplier

We can finally compare the nominal and surrogate adjoint equations [12]

$$\lambda_t = b + \int_0^t A_\tau \lambda_\tau \mathrm{d}\tau, \quad \hat{\lambda}_t = \hat{b} + \int_0^t \hat{A}_\tau \hat{\lambda}_\tau \mathrm{d}\tau,$$

with

$$\begin{aligned}
A_t &= \frac{\partial \bar{f}_\theta(s_t)}{\partial s_t} = \frac{\partial f}{\partial s_t} + \frac{\partial f}{\partial \mu_\theta}\frac{\partial \mu_\theta}{\partial s_t}, \quad b = \frac{\partial R(s_T)}{\partial s_T}, \\
\hat{A}_t &= \frac{\partial \bar{\gamma}_{\omega,\theta}(\hat{s}_t)}{\partial \hat{s}_t} = \frac{\partial \gamma_\omega}{\partial \hat{s}_t} + \frac{\partial \gamma_w}{\partial \mu_\theta}\frac{\partial \mu_\theta}{\partial \hat{s}_t}, \quad \hat{b} = \frac{\partial R(\hat{s}_T)}{\partial \hat{s}_T},
\end{aligned}$$

and compute the error $\tilde{\lambda}_t = \lambda_t - \hat{\lambda}_t$. We have:

$$\tilde{\lambda}_t = \tilde{b} + \int_0^t \left[A_\tau \lambda_\tau - \hat{A}_\tau \hat{\lambda}_\tau\right] \mathrm{d}\tau \tag{6}$$

where $\tilde{b} = b - \hat{b} = \partial_s R(s_T) - \partial_{\hat{s}} R(\hat{s}_T)$. It holds,

$$\|\tilde{b}\| \leq m_{\partial J} \|\tilde{s}_T\| \leq m_{\partial J} \delta_f e^{m_f T}.$$

Next, consider the term under integral sign in the rhs of (6),

$$\begin{aligned}
&A_t \lambda_t - \hat{A}_t \hat{\lambda}_t = \\
&[A_t - \hat{A}_t]\lambda_t + \hat{A}_t \tilde{\lambda}_t = \\
&\underbrace{\left[\mathsf{D}\bar{f}_\theta(s_t) - \mathsf{D}\bar{f}_\theta(\hat{s}_t)\right]\lambda_t}_{\star} - \underbrace{\mathsf{D}\bar{\varepsilon}_\theta(\hat{s}_t)\lambda_t}_{\star} + \underbrace{\hat{A}_t \tilde{\lambda}_t}_{\star} =
\end{aligned}$$

The norm of the first term of the sum can be bounded as

$$\|\star\| \leq \|\mathsf{D}\bar{f}_\theta(s_t) - \mathsf{D}\bar{f}_\theta(\hat{s}_t)\| \|\lambda_t\| \leq m_{\partial f} \delta_f e^{m_f t} \|\lambda_t\|.$$

Further, the norm of the second and third terms can be bounded as

$$\|\star\| \leq m_\varepsilon \|\lambda_t\|, \quad \|\star\| \leq m_\gamma \|\tilde{\lambda}_t\|$$

Taking the norm of (6) we thus obtain

$$\begin{aligned}
\|\tilde{\lambda}_t\| \leq \overbrace{m_{\partial J}\delta_f e^{m_f T}}^{d} + \int_0^t m_\gamma \|\tilde{\lambda}_\tau\| \mathrm{d}\tau \\
\underbrace{+ \int_0^t (m_{\partial f}\delta_f e^{m_f \tau} + m_\varepsilon)\|\lambda_\tau\| \mathrm{d}\tau}_{c_t}
\end{aligned}$$

Using

$$\|\lambda_t\| \leq \|b\| + \int_0^t \|A_\tau\| \|\lambda_\tau\| \mathrm{d}\tau \Rightarrow \|\lambda_t\| \leq m_J e^{m_f t},$$

yields to

$$\begin{aligned}
c_t &= \int_0^t (m_{\partial f}\delta_f e^{m_f\tau} + m_\varepsilon) m_J e^{m_f\tau}\mathrm{d}\tau \\
&= m_J m_{\partial f}\delta_f \int_0^t e^{2m_f\tau}\mathrm{d}\tau + m_J m_\varepsilon \int_0^t e^{m_f\tau}\mathrm{d}\tau \\
&= \frac{m_J m_{\partial f}\delta_f}{2m_f}(e^{2m_f t} - 1) + \frac{m_J m_\varepsilon}{m_f}(e^{m_f} - 1) \\
&= \frac{m_J}{2m_f}[m_{\partial f}\delta_f(e^{2m_f t} - 1) + 2m_\varepsilon(e^{m_f} - 1)].
\end{aligned}$$

Finally,

$$\|\tilde{\lambda}_t\| \leq d + c_t + \int_0^t m_\gamma \|\tilde{\lambda}_\tau\|\mathrm{d}\tau.$$

Since $d + c_t$ is non-decreasing in t (c_t is the integral of a non-negative function), Gronwall's Lemma reads

$$\|\tilde{\lambda}_t\| \leq (d + c_t)e^{m_\gamma t} = \kappa_t.$$

The term $d = m_{\partial J}\delta_f e^{m_f T}$ bounds the error $\|\tilde{b}\| \leq d$ caused by propagating discrepancies in the final state $\tilde{s}_T$ to the objective $R(s_T)$ and its partial derivative $\partial R/\partial s_T$. For example, if $R(s_T) = \frac{1}{2}s_T^\top Q s_T$, We have $\tilde{b} = Q\tilde{s}_T$ and $m_{\partial J} = \sup_{\|s\|=1}\|Qs\|$.

IV. Discussion

This manuscripts introduced preliminary investigations on surrogate models that learn the continuous dynamics of the environment. In practice, the surrogate model serves as a proxy to the policy gradients.

As such surrogate model converges to the actual dynamics of the system, i.e. $\delta_f \to 0 \quad$ and $\quad m_\epsilon \to 0$, the parameter c_t converging to zero. Thus, the error of the Lagrange multiplier also goes to zero, $\forall t \in \mathbb{T} \quad \lim_{|\bar{f}_\theta - \bar{\gamma}_{\omega,\theta}| \to 0} \kappa_t = 0$. Besides, the error grows exponentially with the the length of the time horizon.

We are currently investigating model-based reinforcement learning for continuous stochastic dynamics [42] and SDE-driven stochastic policies. Interesting directions in this space include studying propagation of uncertainties on the objective function, integration time (episode lenght) as well as characterize solutions under partial observability of the state and reachability of the surrogate model.

References

[1] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*, pp. 387–395, PMLR, 2014.

[2] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[3] D. P. Bertsekas *et al.*, *Dynamic programming and optimal control: Vol. 1*. Athena scientific Belmont, 2000.

[4] M. Lutter, S. Mannor, J. Peters, D. Fox, and A. Garg, "Value iteration in continuous actions, states and time," *arXiv preprint arXiv:2105.04682*, 2021.

[5] L. S. Pontryagin, *Mathematical theory of optimal processes*. CRC press, 1987.

[6] X. Zhou, "Maximum principle, dynamic programming, and their connection in deterministic control," *Journal of Optimization Theory and Applications*, vol. 65, no. 2, pp. 363–373, 1990.

[7] D. Q. Mayne and H. Michalska, "Receding horizon control of nonlinear systems," in *Proceedings of the 27th IEEE Conference on Decision and Control*, pp. 464–465, IEEE, 1988.

[8] W. H. Kwon and S. H. Han, *Receding horizon control: model predictive control for state models*. Springer Science & Business Media, 2006.

[9] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer science & business media, 2013.

[10] H. G. Bock and K.-J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.

[11] S. Massaroli, M. Poli, S. Sonoda, T. Suzuki, J. Park, A. Yamashita, and H. Asama, "Differentiable multiple shooting layers," *arXiv preprint arXiv:2106.03885*, 2021.

[12] Y. Cao, S. Li, L. Petzold, and R. Serban, "Adjoint sensitivity analysis for differential-algebraic equations: The adjoint dae system and its numerical solution," *SIAM journal on scientific computing*, vol. 24, no. 3, pp. 1076–1089, 2003.

[13] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter, "End-to-end differentiable physics for learning and control," *Advances in neural information processing systems*, vol. 31, 2018.

[14] P. Holl, V. Koltun, and N. Thuerey, "Learning to control pdes with differentiable physics," *arXiv preprint arXiv:2001.07457*, 2020.

[15] P. Holl, V. Koltun, K. Um, and N. Thuerey, "phiflow: A differentiable pde solving framework for deep learning via physical simulations," in *NeurIPS Workshop*, 2020.

[16] Y.-L. Qiao, J. Liang, V. Koltun, and M. C. Lin, "Scalable differentiable physics for learning and control," *arXiv preprint arXiv:2007.02168*, 2020.

[17] J. Xu, V. Makoviychuk, Y. Narang, F. Ramos, W. Matusik, A. Garg, and M. Macklin, "Accelerated policy learning with parallel differentiable simulation," *arXiv preprint arXiv:2204.07137*, 2022.

[18] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[19] M. Poli, S. Massaroli, A. Yamashita, H. Asama, and J. Park, "Torchdyn: A neural differential equations library," *arXiv preprint arXiv:2009.09346*, 2020.

[20] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," *arXiv preprint arXiv:1806.07366*, 2018.

[21] P. Kidger, "On neural differential equations," *arXiv preprint arXiv:2202.02435*, 2022.

[22] W. Grathwohl, R. T. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, "Ffjord: Free-form continuous dynamics for scalable reversible generative models," *arXiv preprint arXiv:1810.01367*, 2018.

[23] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan, and A. Edelman, "Universal differential equations for scientific machine learning," *arXiv preprint arXiv:2001.04385*, 2020.

[24] S. Massaroli, M. Poli, J. Park, A. Yamashita, and H. Asama, "Dissecting neural odes," *arXiv preprint arXiv:2002.08071*, 2020.

[25] Y. Rubanova, R. T. Chen, and D. K. Duvenaud, "Latent ordinary differential equations for irregularly-sampled time series," *Advances in neural information processing systems*, vol. 32, 2019.

[26] A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Ré, "Hippo: Recurrent memory with optimal polynomial projections," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1474–1487, 2020.

[27] J. T. Smith, A. Warrington, and S. W. Linderman, "Simplified state space layers for sequence modeling," *arXiv preprint arXiv:2208.04933*, 2022.

[28] J. Du, J. Futoma, and F. Doshi-Velez, "Model-based reinforcement learning for semi-markov decision processes with neural odes," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19805–19816, 2020.

[29] C. Yildiz, M. Heinonen, and H. Lähdesmäki, "Continuous-time model-based reinforcement learning," in *International Conference on Machine Learning*, pp. 12009–12018, PMLR, 2021.

[30] L. Sforni, I. Notarnicola, and G. Notarstefano, "Learning-driven nonlinear optimal control via gaussian process regression," in *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 4412–4417, IEEE, 2021.

[31] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho, "Lagrangian neural networks," *arXiv preprint arXiv:2003.04630*, 2020.

[32] S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian neural networks," *Advances in neural information processing systems*, vol. 32, 2019.

[33] S. Massaroli, M. Poli, F. Califano, J. Park, A. Yamashita, and H. Asama, “Optimal energy shaping via neural approximators,” *arXiv preprint arXiv:2101.05537*, 2021.

[34] M. Deisenroth and C. E. Rasmussen, “Pilco: A model-based and data-efficient approach to policy search,” in *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472, Citeseer, 2011.

[35] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, “Gaussian processes for data-efficient learning in robotics and control,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 2, pp. 408–423, 2013.

[36] D. Ha and J. Schmidhuber, “World models,” *arXiv preprint arXiv:1803.10122*, 2018.

[37] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, “Decision transformer: Reinforcement learning via sequence modeling,” *Advances in neural information processing systems*, vol. 34, pp. 15084–15097, 2021.

[38] Q. Zheng, A. Zhang, and A. Grover, “Online decision transformer,” *arXiv preprint arXiv:2202.05607*, 2022.

[39] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, *et al.*, “A generalist agent,” *arXiv preprint arXiv:2205.06175*, 2022.

[40] G. Peano, *Sull’integrabilità delle equazioni differenziali di primo ordine*. Loescher, 1886.

[41] T. H. Gronwall, “Note on the derivatives with respect to a parameter of the solutions of a system of differential equations,” *Annals of Mathematics*, pp. 292–296, 1919.

[42] S. Massaroli, M. Poli, S. Peluchetti, J. Park, A. Yamashita, and H. Asama, “Learning stochastic optimal policies via gradient descent,” *IEEE Control Systems Letters*, 2021.