Topics on Differentiable Simulation

Jing Yuan Luo

December 13, 2023

1 Literature Review

Brief write-ups on all underactuated robotics chapters [1] [2]

Vision-Based Locomotion: - Underactuated robotics pixels to torque chapter: there's a lack of results. - Using deep RL - No-one's done it on a non-trivial system yet

Analytical Policy Gradients: - Brief lit review. Ex: RL search for LQR. - Suh contact rich ... - RL from the perspective of continuous control - Na Li ...

2 Very Small Policy

We work with a 12-DoF quadruped (3 per limb) in a discrete simulator. For Table 1, N = 18, M = 12, D = 48. The idea is to have a locomotion policy with very few parameters, by learning how to go in the desired directions. Since all joints are revolute, their coordinates are simply scalar radians.

System:

$$u_k = \pi(x_k, \theta) \tag{1}$$

$$x_{k+1} = f(x_k, u_k) \tag{2}$$

Policy: Each limb j tracks a 4-parameter sine reference using a PD controller.

$$u_k = \pi(x_k, \theta) = K(r(k;\theta) - q_{J,k}) + K_d(-\dot{q}_{J,k}) \tag{3}$$

$$r(k;\theta)_j = a_j \sin(2\pi \mathrm{dt} s_j k + \psi_j) + b_j \tag{4}$$

$$\theta_j \doteq [a_j, s_j, \psi_j, b_j] \text{ for } j \in \{1, ..., M\} \tag{5}$$

Problem Formulation:

$$\min_{\theta} L(\theta, S) \tag{6}$$

$$L(\theta, S) = \sum_{k=0}^{K-1} \frac{1}{2} ||S(x^* - x_{k+1})||^2 \tag{7}$$

Symbol	
k	timestep
K	timesteps per gradient update
x	full state; $[q^T, \dot{q}^T]^T \in \mathbb{R}^{2N}$
u	action $\in \mathbb{R}^M$
θ	policy parameters $\in \mathbb{R}^D$
π	policy: $\mathbb{R}^N \times \mathbb{R}^D \to \mathbb{R}^M$
f	simulation step: $\mathbb{R}^N \times \mathbb{R}^M \to \mathbb{R}^N$
q	vector of coordinates: $[q_B^T, q_J^T]^T$
q_B	vector of floating base coordinates $\in \mathbb{R}^6$
q_J	vector of joint coordinates $\in \mathbb{R}^M$
$\dot{q}$	vector of velocities: $[\dot{q}_B^T, \dot{q}_J^T]^T$
r_k	reference position at timestep k $\in \mathbb{R}^M$
K	diagonal matrix of P-gains $\in \mathbb{R}^{M \times M}$
K_D	diagonal matrix of D-gains $\in \mathbb{R}^{M \times M}$

Table 1: Notation

Where $S \in \mathbb{R}^{N \times N}$ is a (weighted) selection matrix for the dimensions to penalize for error and q_B^* is the desired end position. For forward locomotion, by symmetry of the quadruped, we can constrain the parameters, reducing θ from $\mathbb{R}^{48}$ to $\mathbb{R}^5$:

1. Static hips: $a_j = 0$ for all four hip joints.
2. *Synchronization Constraints*
   (a) Using ψ_j: thighs and shanks have a $\pi/2$ phase offset. Opposite pairs of legs have a π offset.
   (b) $s^i = s^j$ for all i, j. (fixed period scale)

Analytical Policy Gradient: *wlog*, assume S is $\mathbb{I}_{2N \times 2N}$. We can make this assumption because for any k, $||S(x^* - x_{k+1})||^2 = (x^* - x_{k+1})^T S^T S (x^* - x_{k+1})$. Since $S^T S = S$, we simply expand out the quadratic form: $\sum_{i \in \mathcal{S}} (x_i^* - x_{k+1,i})^2$ where $\mathcal{S}$ is the set of indices where the diagonal entry of S is 1. This sum is simply $(x_{\mathcal{S}}^* - x_{\mathcal{S},k+1})^T \mathbb{I} (x_{\mathcal{S}}^* - x_{\mathcal{S},k+1})$, preserving the form resulting from the above assumption.

Denote $l_k(\theta) \doteq 1/2||(x^* - x_{k+1})\mathbb{I}(x^* - x_{k+1})||$. Assume that the partial derivatives $\frac{\partial}{\partial \theta_m} x_{k+1,l}$ exist everywhere for all m and l. Then, the total derivatives are equivalent to the jacobians and we can apply the chain rule. All terms obtainable from the differentiable simulation are green, those from previous timesteps are blue.

$$\nabla_\theta \sum_{k=0}^{K-1} \frac{1}{2}||\mathbb{I}(x^* - x_{k+1})||^2 = \sum_{k=0}^{K-1} \frac{1}{2}\nabla_\theta l_k \tag{8}$$

$$\nabla_\theta l_k = 2(x^* - x_{k+1})^T \nabla_\theta f(x_k, u_k) \tag{9}$$

$$\nabla_\theta (f(x_k, u_k)) = \nabla_{x_k} f(x_k, u_k) \nabla_\theta x_k + \nabla_{u_k} f(x_k, u_k) \nabla_\theta u_k \tag{10}$$

$$\nabla_\theta u_k = \nabla_r u_k \nabla_\theta r + \nabla_{q_J} u_k \nabla_\theta q_J \tag{11}$$

$$= K(\nabla_\theta r - \nabla_\theta q_J) \tag{12}$$

$$\nabla_\theta r(k;\theta)_{j,I(j)} = \begin{bmatrix} sin(2\pi \mathrm{dt} s_j k + \psi_j) \\ a_j cos(2\pi \mathrm{dt} s_j k + \psi_j) 2\pi \mathrm{dt} k \\ a_j cos(2\pi \mathrm{dt} s_j k + \psi_j) \\ 1 \end{bmatrix} \tag{13}$$

Where steps (10) and (11) follow from the multivariate chain rule and steps (9) and (12) follow from matrix calculus. In step (13), j corresponds to joint indies and $I(j)$ designates the sequential indicies of the parameters corresponding joint j: $I(j) : j \mapsto \{4(j-1)+1, ..., 4(j-1)+4\}$.

Note: Select particular states in the loss function by extracting the relevant rows of the error vector and the gradient calculated in step (10). □

Algorithm:

Let $\bar{\theta}$ be the unique parameters resulting from the constraints, indexed by $p \in \{1, ..., P\}$. Let $\mathcal{I}(p)$ be the set of indicies i of θ that go into p. Note that we can update the reference x^* in different epochs, to achieve different positions.

Algorithm 1: Constrained online gradient descent

Initialisation: Initialize x^*, x_0, θ_0
for $e = 1, 2, ..., \infty$ **do**
  $x^*, S \leftarrow$ update(e)
  Store K samples $\{x_k, u_k, \nabla_x f(x_k, u_k), \nabla_u f(x_k, u_k)\}_{k=0}^{K-1}$
  $\theta_{k+1} \leftarrow (1-\alpha)\theta_k + \alpha \nabla_\theta L(\theta_k, S)$
  for *p = 1, ..., P* **do**
    $\bar{\theta}_p \leftarrow \frac{1}{\mathcal{I}(p)} \sum_{l \in \mathcal{I}(p)} \theta_{k+1,l}$
    $\theta_{k+1,l} \leftarrow \bar{\theta}_p$ for all $l \in \mathcal{I}(P)$
  end
end

3 Gradients

3.1 Linearized Models

Because of the our stronger understanding of linear systems, it is often worthwhile to work on linearized versions of non-linear systems. Specifically, say we

linearize at timestep l, and try to predict the state at timestep $k+1$. Taylor's multivariate theorem implies (under a relaxed set of technical assumptions):

$$f(x_k, u_k) = f(x_l, u_l) + J_x f(x_l, u_l)\Delta x + J_u f(x_l, u_l)\Delta u + \tag{14}$$

$$\mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta u^2) + \mathcal{O}(\Delta x \Delta u) \tag{15}$$

$$\text{Where } \Delta x \doteq x_k - x_l, \Delta u \doteq u_k - u_l \tag{16}$$

$$[J_x f(x_l, u_l)]_{i,j} \doteq \frac{\partial f_i(x_l, u_l)}{\partial x_j} \tag{17}$$

$$[J_u f(x_l, u_l)]_{i,n} \doteq \frac{\partial f_i(x_l, u_l)}{\partial u_n} \tag{18}$$

$$\tag{19}$$

Clearly, the linearization is practical if and only if the error terms (15) show good scaling properties.

3.2 Linearizing Models with Orientation State

Representing Orientation

While the translational orientation and angular velocity of a system are quantities easily represented in $\mathbb{R}^3$, representing its angular orientation is far less trivial.

Orientation is typically represented (via a *parameterization*) with respect to a fixed "Newtonian" or "Inertial" frame. These parameterizations vary greatly in size and mathematical properties. A result by Euler indicates that any orientation can be represented by a 3D axis of rotation (typically normalised) and a rotation amount around that axis. There are many 3-parameter representations of orientation, most of which can be understood through this intuition.

However, it is well-known that all 3-parameter representations suffer from *singularities*. As a simple example, consider the commonly used Rodriguez parameterization. Let $\boldsymbol{n}$ be the normalised euler angle and θ be the rotation amount. Then, this representation can be written as:

$$\tan\left(\frac{\theta}{2}\right)\boldsymbol{n} \tag{20}$$

Since $\tan(\cdot) = \frac{\sin(\cdot)}{\cos(\cdot)}$, we see that the paramterization is infinite at $\theta = \pi$. The weakness of this particular parameterization is that it can only be used to represent small deviations in orientation.

Unit quaternions have unit L2-norm. They represent orientation using four parameters and do not suffer from singularities. They can be written as:

$$\begin{pmatrix} \cos(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2})\boldsymbol{n} \end{pmatrix} \tag{21}$$

A key property of unit quaternions is that one can define a simple multiplication primitative between them that is both closed (the result is also a unit quaternion) and associative (a rotation from l to m with a rotation from m to n equals one from l to n). Let $\boldsymbol{v}$ and s be the vector and scalar parts of the unit quaternion; $\boldsymbol{v} \doteq \sin(\frac{\theta}{2})\boldsymbol{n}, s = \cos(\frac{\theta}{2})$.

This primitive is as follows:

$$q^a \circ q^b = L(q^a)q^b \tag{22}$$

$$L(q) = \begin{bmatrix} s & & -\boldsymbol{v}^T & \\ & s & v_3 & -v_2 \\ \boldsymbol{v} & -v_3 & s & v_1 \\ & v_2 & -v_1 & s \end{bmatrix} \tag{23}$$

Where 22 indicates the standard matrix-vector product.

Note that for several 3-dimensional orientation parameterizations, conversion to and from unit quaternions is simple. For instance, the rodriguez parameterization of a unit quaternion is simply $\boldsymbol{r} = \boldsymbol{v}/s$. To return to a unit quaternion, one simply normalizes the vector $[1\boldsymbol{r}]$.

Attitude Jacobians

A robust way to linearize models with quaternion orientation is the method of attitude jacobians. Attitude jacobians are motivated by two observations. First, small differences in orientation do not correspond the the mathematical operation of coordinate subtraction that is usually used in first-order methods. For instance, the taylor series of a scalar function x would be:

$$f(x) \approx x_l + \frac{\partial f}{\partial x}(x - x_l) \tag{24}$$

With $\Delta x = x - x_l$ representing the difference between the linearization point x_l and x. Say we replaced x with q_k; $\Delta q_k \doteq q_k - q_l$ is *not* a unit quaternion representation of the orientation between q_l and q. Indeed, defining ${}_a q_b$ as the orientation of some frame b relative to frame a, Δq_k should instead be written as ${}_l q_k$ such that ${}_I q_k = {}_I\, q_l \circ_l q_k$. Here, I denotes the inertial frame. Going forward, we use $\tilde{q}$ to for small quaternion orientation differences.

Second, if we use this method to represent small differences in orientation, the *locality* of the first order approximation may not hold. In the above scalar example, $\Delta x \in \mathbb{R}$ can be arbitrarily small. In comparison, $||\tilde{q}|| = 1\ \forall q \in Q$, where Q is the space of unit quaternions. Note that there are other orientation representations that can have arbitrary norms. For instance, the rodriguez parameterization (eqn. 20) shrinks to zero norm for small orientation changes. It captures both the the mathematics of the taylor series and the representation of relative orientation.

The attitude jacobian leverages the first observation by working with "legal" representations of small orientation changes $\tilde{q}$, and by doing the first-order

approximations using the rodriguez parameterization. We will denote $\Phi(q)$ as the transformation from the quaternion to the rodriguez parameterization of the same orientation.

We now define the **Attitude Jacobian**:

$$G(q) = L(q)H \tag{25}$$

$$H = \begin{bmatrix} \mathbf{1}_{1x3} \\ \mathbb{I}_{3x3} \end{bmatrix} \tag{26}$$

$$G(q) \in \mathbb{R}^{4\times 3} \tag{27}$$

Now, let $f_1 : Q \to \mathbb{R}^p$ be a function mapping from the space of unit quaternions to p-dimensional real numbers, and $f_2 : Q \to Q$ map between unit quaternions. It can be shown that to a first-order approximation:

$$f_1(q) \approx f_1(q_l) + J_q f_1(q_l) G(q_l)\tilde{\boldsymbol{r}} \tag{28}$$

$$f_2(q) \approx f_2(q_l) \circ \Phi^{-1}(G(q_{l+1})^T J_q f_2(q_l) G(q_l)\tilde{\boldsymbol{r}}) \tag{29}$$

$$q_{l+1} \doteq f_2(q_l) \tag{30}$$

$$\tag{31}$$

Note that $\tilde{\boldsymbol{r}}$ is the rodriguez representation of the orientation difference from q_l to q, i.e. $\tilde{\boldsymbol{r}} = \Phi(\tilde{q})$. Note also:

$$\tilde{q} = L^T(q_l)q \tag{32}$$

We now have the ingredients to define the attitude jacobian method.

These modified A and B matricies represent the linearized system dynamics. For instance, applied to the Linear Quadratic Regulator problem, the output policy matrix of Ricatti Recursion performs remarkably well in stabilizing out disturbances in orientation state [TODO: Cite Manchester lectures].

3.3 Quadruped dynamics are highly non-linear

Linearized Optimal Control

Practical schemes to control non-linear systems often involve linearizing them at known points. This is used, for instance, in Linear MPC, in which an expensive trajectory optimisation is performed offline, resulting in $\bar{x}_i, \bar{u}_i{}_{i=1}^{T}$. T linearizations are performed around these points, and a linear MPC problem is solved using them. This has the advantage of quick and reliable online solving - if the engineer selects a cost function and constraints such that the problem is quadratic, it can typically be solved in one step by mature solvers such as Ipopt.

Algorithm 2: Linearized Fixed Reference Tracking

Initialisation: Initialize x^*, x_1, u_1
for $k = 1, 2, ..., \infty$ **do**
  $A \leftarrow J_x f(x_k, u_k)$
  $B \leftarrow J_u f(x_k, u_k)$
  $b \leftarrow f(x_k, u_k)$
  $\{u_\kappa, x_\kappa\}_{\kappa=k+1}^{k+H}$ = LinearMPCTracking(A, B, b, x_k, u_k, x^*, x_k)
  apply u_{k+1}
  measure x_{k+1}
end

LinearMPCTracking (LMT) is defined as:

$$LMT(A, B, b, x_{lin}, u_{lin}, x^*, x_{init}) = \tag{33}$$

$$\min_{\substack{x_k, u_k \\ k=1,\dots,H}} (x_H - x^*)^T Q_1 (x_H - x^*) + \sum_{k=1}^{H-1} (x_k - x^*)^T Q_1 (x_k - x^*) + \alpha u_k^T Q_2 u_k \tag{34}$$

$$\text{subject to:} \tag{35}$$

$$u_{\mathcal{B}} = 0 \tag{36}$$

$$x_1 = x_{init} \tag{37}$$

$$x_H = x^* \tag{38}$$

$$x_{k+1} = b + A(x_k - x_{lin}) + B(u_k - u_{lin}) \tag{39}$$

$$k = 1, ..., H \tag{40}$$

Q1 and Q2 are square $N \times N$ matricies set to identity, α sets the penalty for large controls and is tuned. The dynamics constraints (39) are a first-order approximation, using jacobians given by a differentiable simulator.

Gradient Analysis

Linearization techniques are not easily applicable to Anymal. For instance, the above straight-forward algorithm fails for the simple control task of standing. This section indicates empirical evidence as to why this may be the case - the combination of the task and and robot is so non-linear that the approximation guarantees from linearization are within a region so small as to be impractical.

Setup

We apply a simple PD controller separately to each joint, with gains of K = 150 and K_d = 1.5. We initialize the Anymal at $x_1 = x^*$, and the controller achieves near-perfect stabilization at the target x^* after some initial transients. In this analysis, we observe the states of the right front (RF) leg. These were observed to be the same as those of the other legs, and are more relevant to control than the states of the robot body.

Proceeding, we focus on the state of the knee joint, which was observed to have the largest oscillations. We implement the distance Δ by using states and control inputs from d timesteps ago - the larger d is, the larger the state. We chose this rather than fixed Δ's due to the implementation of our differential simulator and because Δ being approximated by timestep delay is a common scenario in practice.

To understand the relation between time step delay and the distance $\Delta_x = \Delta x_k - \Delta x_{k-d}$, see the following:

Growth of Error Terms

4 Low-level Control

Position Tracking using Inverse Dynamics

$$\tau^d = M_j(q)\dot{u}^* + h_j(q,u) - J_{s,j}(q)\lambda^* \tag{41}$$

$$\tau^{ref} = \tau^d + k_P\tilde{q} + k_D\dot{\tilde{q}} \tag{42}$$

References

[1] R. Featherstone, *Rigid Body Dynamics Algorithms*. Boston, MA: Springer US, 2008, ISBN: 978-0-387-74314-1. DOI: 10.1007/978-1-4899-7560-7. [Online]. Available: http://link.springer.com/10.1007/978-1-4899-7560-7 (visited on 12/13/2023).

[2] B. E. Jackson, K. Tracy, and Z. Manchester, "Planning With Attitude," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5658–5664, Jul. 2021, ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2021.3052431. [Online]. Available: https://ieeexplore.ieee.org/document/9326337/ (visited on 12/13/2023).