An Introduction to iLQR/DDP

DIPARTIMENTO DI
INGEGNERIA
DELL'INFORMAZIONE

Research Center E. Piaggio
University of Pisa

OUTLINE:

- Solution methods for LQR problem
    - LQR via Shooting
    - LQR via QP
    - LQR via Riccati Recursion
- iLQR/DDP introduction
- Some recent works

LQR

$$\min_{\mathbf{x},\mathbf{u}} J = \sum_{n=1}^{N-1} \frac{1}{2}(x_n)^T Q_n (x_n) + \frac{1}{2}(u_n)^T R_n (u_n) + \frac{1}{2}(x_N)^T Q_N (x_N)$$

$$\text{s.t.} \quad x_{n+1} = A\, x_n + B\, u_n$$

The LQR problem:

1. It can locally approximate many nonlinear systems
2. It is computationally tractable
3. Many possible extensions are already available e.g. infinite horizon, stochastic, robust

Let’s dive into solution methods for LQR

LQR via Indirect Shooting

$$\min_{\mathbf{x},\mathbf{u}} J = \sum_{n=1}^{N-1} \frac{1}{2}(x_n)^T Q_n (x_n) + \frac{1}{2}(u_n)^T R_n (u_n) + \frac{1}{2}(x_N)^T Q_N (x_N)$$

$$\text{s.t.} \quad x_{n+1} = A\, x_n + B\, u_n$$

define $\mathbf{X} \triangleq (x_1, x_2, .., x_N)$ and $\mathbf{U} \triangleq (u_1, u_2, .., u_N)$

Then applying the PMP principle

$$\begin{aligned} x_{n+1} &= \nabla_\lambda H(x_n, u_n, \lambda_{n+1}) = Ax_n + Bu_n \\ \lambda_n &= \nabla_x H(x_n, u_n, \lambda_{n+1}) = Qx_n + A^T \lambda_{n+1} \\ \lambda_N &= Q_N x_N \\ \nabla u_n &= \underset{\tilde{u}}{\operatorname{argmin}}\ H(x_n, \tilde{u}, \lambda_{n+1}) = -R^{-1} B^T \lambda_{n+1} \end{aligned}$$

Procedure

1. Start with an initial guess trajectory
2. Simulate ("rollout") to get x
3. Backward pass to get λ and δ_u
4. Rollout with line search on δ_u
5. Goto 3 until convergence

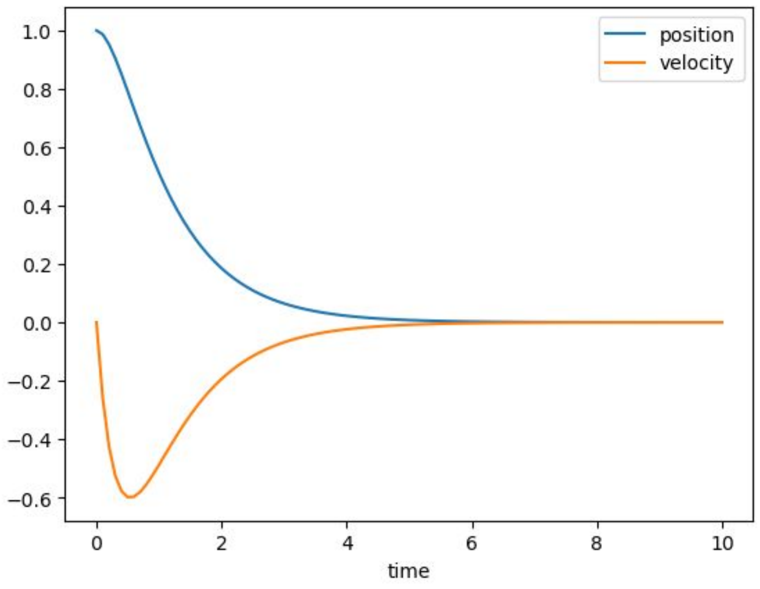
Example: Double Integrator

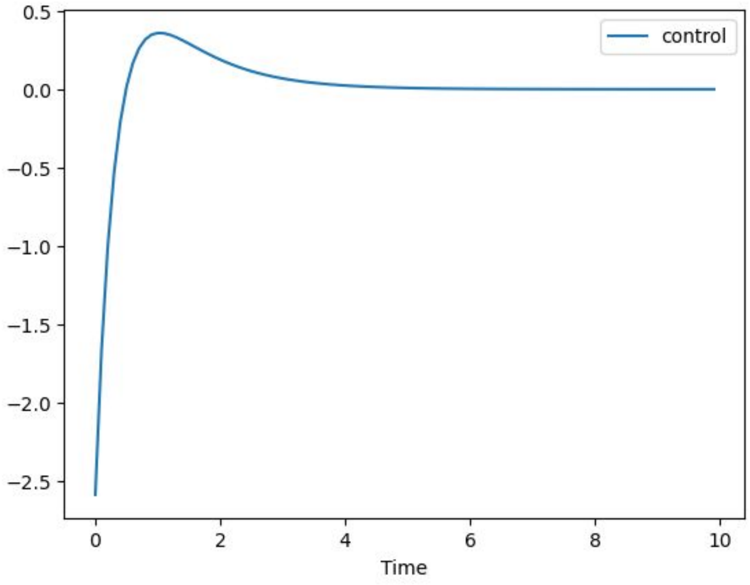
$$\dot{x} = \begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

Discrete version using Euler integration

$$x_{n+1} = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} \begin{bmatrix} q_n \\ \dot{q}_n \end{bmatrix} + \begin{bmatrix} \frac{1}{2}h^2 \\ h \end{bmatrix} u_n$$

Simulation results





LQR via Quadratic Programming (QP)

- Assume the initial state x_0 is given (so it is not a decision variable)
- Let's condense the problem using the following definition

$$z = \begin{bmatrix} u_1 \\ x_2 \\ u_2 \\ . \\ . \\ . \\ x_N \end{bmatrix} \quad and \quad H = \begin{bmatrix} R_1 & & & & \\ & Q_2 & & & \\ & & R_2 & & \\ & & & \ddots & \\ & & & & Q_N \end{bmatrix}$$

Such that $$J = \frac{1}{2} z^T H z$$

Similarly the equality constraints (representing the dynamics equation) can also be condensed into:

$$\begin{bmatrix} B & -I & 0 & 0 & \cdot & \cdot & \\ 0 & A & B & -I & 0 & \cdot\cdot & \\ & & \ddots & & & & \\ & & & \ddots & & & \\ & & & & A_{N-1} & B_{N-1} & -I \end{bmatrix} \begin{bmatrix} u_1 \\ x_2 \\ u_2 \\ \cdot \\ \cdot \\ \cdot \\ x_N \end{bmatrix} = \begin{bmatrix} Ax_1 \\ 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}$$

Thus the original optimisation problem can be recast into a condensed form as:

$$\min_z \; \frac{1}{2} z^T H z$$
$$\text{s.t. } Cz = d$$

The lagrangian of the problem thus is:

$$L(z, \lambda) = \frac{1}{2} z^T H z + \lambda^T [Cz - d]$$

KKT conditions

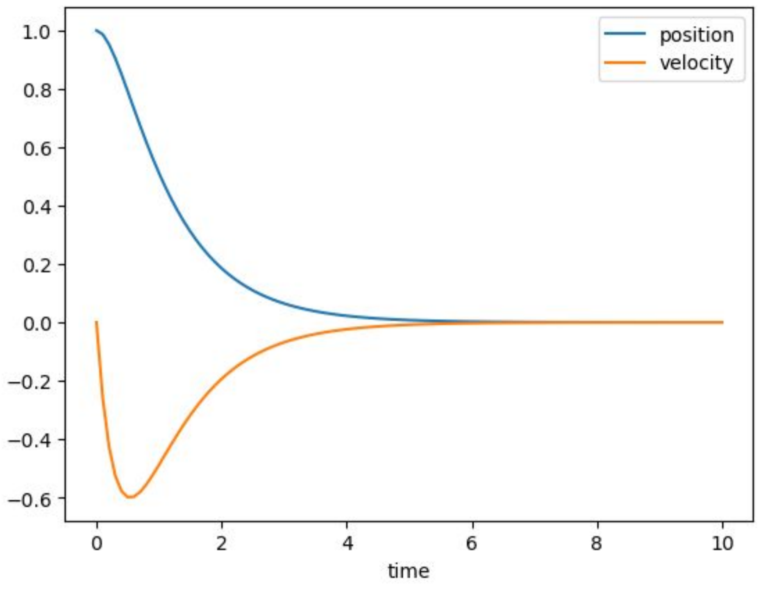
$$\nabla_z L = \; Hz + \; C^T \lambda = 0$$

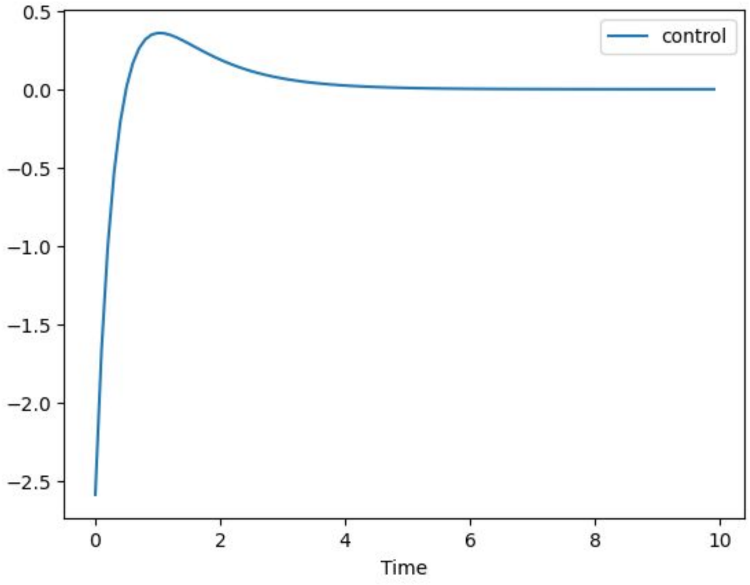
$$\nabla_\lambda L = \; Cz - d = 0$$

implies

$$\begin{bmatrix} H & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} z \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ d \end{bmatrix} \qquad \text{Eq. (2)}$$

Simulation Results (Double Integrator example)





We get the similar results by solving one linear system!

Riccati Recursion

Let’s look deeply into the Eq. 2 in the previous slides.

$$
\left[\begin{array}{cccccc|ccc}
R & & & & & & B^T & & \\
& Q & & & & & -I & A^T & \\
& & R & & & & & B^T & \\
& & & Q & & & & -I & A^T \\
& & & & R & & & & B^T \\
& & & & & Q_4 & & & -I \\
\hline
B & -I & & & & & & & \\
& A & B & -I & & & & 0 & \\
& & & A & B & -I & & &
\end{array}\right]
\begin{bmatrix} u_1 \\ x_2 \\ u_2 \\ x_3 \\ u_3 \\ x_4 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{bmatrix}
=
\begin{bmatrix} 0 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ -Ax_1 \\ 0 \\ 0 \end{bmatrix}
\qquad \text{Eq. 3}
$$

Using back substitution trick

$$Q_N x_1 - \lambda_4 = 0 \Rightarrow \lambda_4 = Q_N x_4$$

$$R u_3 + B^T \lambda_4 = R u_3 + B^T Q_n x_4 = 0$$

One can obtain u_3 from the above equation

$$\Rightarrow \quad Ru_3 + B^T Q_N (Ax_3 + Bu_3) = 0$$

$$u_3 = \underbrace{-(R + B^T Q_N B)^{-1} B^T Q_N A}_{k_3} x_3$$

Substituting the u_3 in the equations obtained from one row above in Eq. 3

$$Qx_3 - \lambda_3 + A^T Q_N (Ax_3 + Bu_3) = 0$$

plug in $u_3 = k_3 x_3$

$$\rightarrow \quad \lambda_3 = (\underbrace{Q + A^T Q_N (A - Bk)}_{P_3}) x_3$$

Thus we obtain the following recursion equation:

$$P_N = Q_N$$

$$K_n = (R + B^T P_{n+1} B)^{-1} B^T P_{n+1} A$$

$$P_n = Q + A^T P_{n+1} (A - B K_n)$$

These are similar equations as Bellman Optimality Conditions for LQR!!!

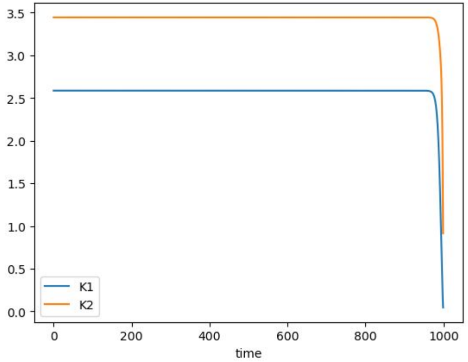
- We can solve the LQR by solving the backward riccati recursion followed by a forward rollout.

- QP has a complexity of $O(N^3(n+m)^3)$
Riccati solution is $O(N(n+m)^3)$

- Most importantly we obtain a feedback policy instead of an open loop trajectory.
So, we can start at any initial point and follow the feedback policy.

- Even in the presence of stochastic noise in the system, the feedback policy can stabilise to the desired point.

The Feedback matrices stabilise for time-invariant systems



Relation between DP and Riccati equations obtained from QP

Recall from the Riccati equations obtained using QP

$$\lambda_n = P_n * x_n$$

Thus $\lambda = \nabla_x V(x)$

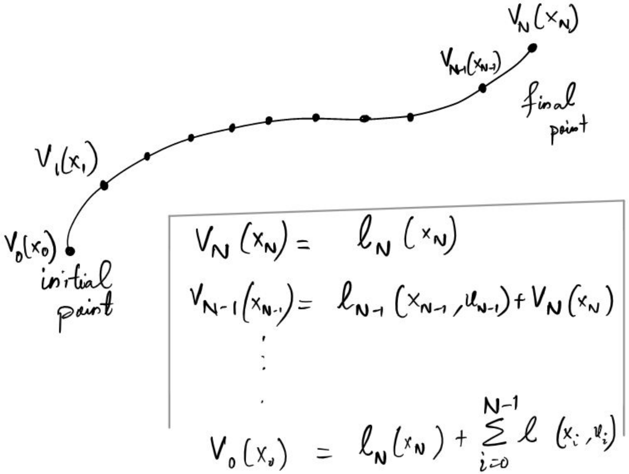
Dynamics lagrange multipliers are Cost-to-go gradients.

It is valid for the nonlinear case as well.

Dynamic Programming

Bellman Optimality Condition:

$$V_i(\mathbf{x}_i) = \min_u \ell(\mathbf{x}_i, \mathbf{u}_i) + V_{i+1}(\mathbf{f}(\mathbf{x}, \mathbf{u}))$$



The optimal solution to the above equation lead to Riccati recursion.

But DP has the curse of dimensionality

Convex MPC

- LQR cannot handle constraints.
- We need MPC to handle constraints.
  - Can solve multistep problems with the terminal cost being obtained from LQR
  - Care must be taken as the size of horizon affects the quality of approximation of terminal cost
- Non-Linear Trajectory optimisation considers non-convex cost with nonlinear dynamics and non-convex constraints.
- The best algorithm in terms of runtime is iLQR/DDP based methods.

Solving OC problems using DP

- Representing the value function is a difficult task. There are various methods of approximating the value function.

- In Deep Reinforcement Learning, generally a function approximator in form of a "Neural network" is used to represent the value function.

- In the following, we will discuss the way of "Taylor expansion" of the value function and then use tools from numerical optimisation.

"Synthesis and Stabilization of Complex Behaviors through Online Trajectory Optimization" Tassa et. al

iLQR/DDP

Bellman Optimality:

Sub-trajectories of the optimal trajectories must be optimal under appropriately defined objective function

$$V(\delta \mathbf{x}_k) = \min_{\delta \mathbf{u}_k} \ell_k(\delta \mathbf{x}_k, \delta \mathbf{u}_k) + V_{k+1}(\mathbf{f}(\delta \mathbf{x}_k, \delta \mathbf{u}_k))$$

Define Q-function as

$$\mathbf{Q}(\delta \mathbf{x}_k, \delta \mathbf{u}_k) = \ell_k(\delta \mathbf{x}_k, \delta \mathbf{u}_k) + V_{k+1}(\mathbf{f}(\delta \mathbf{x}_k, \delta \mathbf{u}_k))$$

Thus,

$$\delta \mathbf{u}_k = \arg\min_{\delta \mathbf{u}_k} \mathbf{Q}(\delta \mathbf{x}_k, \delta \mathbf{u}_k)$$

Now, we use the second order Taylor expansion of the value function and impose the Bellman optimality on the value function.

Note: we linearise around x_k and u_k so we find \delta u_k and \delta x_k in the above equations

Quadratic expansion of Cost-to-go:

$$\Delta V = \min_{\delta \mathbf{u}_k} \frac{1}{2} \begin{bmatrix} \delta \mathbf{x}_k \\ \delta \mathbf{u}_k \end{bmatrix}^\top \begin{bmatrix} \mathbf{Q}_{\mathbf{x}\mathbf{x}_k} & \mathbf{Q}_{\mathbf{x}\mathbf{u}_k} \\ \mathbf{Q}_{\mathbf{u}\mathbf{x}_k} & \mathbf{Q}_{\mathbf{u}\mathbf{u}_k} \end{bmatrix} \begin{bmatrix} \delta \mathbf{x}_k \\ \delta \mathbf{u}_k \end{bmatrix} + \begin{bmatrix} \delta \mathbf{x}_k \\ \delta \mathbf{u}_k \end{bmatrix}^\top \begin{bmatrix} \mathbf{Q}_{\mathbf{x}_k} \\ \mathbf{Q}_{\mathbf{u}_k} \end{bmatrix}.$$

where,

$$\mathbf{Q}_{\mathbf{x}\mathbf{x}_k} = \mathcal{L}_{\mathbf{x}\mathbf{x}_k} + \mathbf{f}_{\mathbf{x}_k}^\top V_{\mathbf{x}\mathbf{x}_{k+1}} \mathbf{f}_{\mathbf{x}_k}, \qquad \mathbf{Q}_{\mathbf{x}_k} = \ell_{\mathbf{x}_k} + \mathbf{f}_{\mathbf{x}_k}^\top V_{\mathbf{x}_{k+1}}^+,$$

$$\mathbf{Q}_{\mathbf{u}\mathbf{u}_k} = \mathcal{L}_{\mathbf{u}\mathbf{u}_k} + \mathbf{f}_{\mathbf{u}_k}^\top V_{\mathbf{x}\mathbf{x}_{k+1}} \mathbf{f}_{\mathbf{u}_k}, \qquad \mathbf{Q}_{\mathbf{u}_k} = \ell_{\mathbf{u}_k} + \mathbf{f}_{\mathbf{u}_k}^\top V_{\mathbf{x}_{k+1}}^+,$$

$$\mathbf{Q}_{\mathbf{x}\mathbf{u}_k} = \mathcal{L}_{\mathbf{x}\mathbf{u}_k} + \mathbf{f}_{\mathbf{x}_k}^\top V_{\mathbf{x}\mathbf{x}_{k+1}} \mathbf{f}_{\mathbf{u}_k},$$

Backward Pass

Evaluate the Jacobians and Hessian of Q-function and then find the control action.

$$\delta \mathbf{u}_k = \underset{\delta \mathbf{u}_k}{\arg\min}\ \mathbf{Q}(\delta \mathbf{x}_k, \delta \mathbf{u}_k) = \hat{\mathbf{k}} + \hat{\mathbf{K}} \delta \mathbf{x}_k$$

Feedforward term: $\hat{\mathbf{k}} = -\mathbf{Q}_{\mathbf{uu}_k}^{-1} \mathbf{Q}_{\mathbf{u}_k}$

Feedback term: $\hat{\mathbf{K}} = -\mathbf{Q}_{\mathbf{uu}_k}^{-1} \mathbf{Q}_{\mathbf{ux}_k}$

$$\mathbf{V}_{\mathbf{x}}(i) = \mathbf{Q}_{\mathbf{x}} + \mathbf{K}^\top \mathbf{Q}_{\mathbf{uu}} \mathbf{k} + \mathbf{K}^\top \mathbf{Q}_{\mathbf{u}} + \mathbf{Q}_{\mathbf{ux}}^\top \mathbf{k},$$
$$\mathbf{V}_{\mathbf{xx}}(i) = \mathbf{Q}_{\mathbf{xx}} + \mathbf{K}^\top \mathbf{Q}_{\mathbf{uu}} \mathbf{K} + \mathbf{K}^\top \mathbf{Q}_{\mathbf{ux}} + \mathbf{Q}_{\mathbf{ux}}^\top \mathbf{K}.$$

Forward Pass

- Rollout the dynamics using the feedback and feedforward terms.

$$\hat{\mathbf{u}}_i = \mathbf{u}_i + \alpha \mathbf{k}_i + \mathbf{K}_i(\hat{\mathbf{x}}_i - \mathbf{x}_i),$$
$$\hat{\mathbf{x}}_{i+1} = \mathbf{f}(\hat{\mathbf{x}}_i, \hat{\mathbf{u}}_i),$$

- Line search using Armijo rule
- Check for convergence
  else repeat Backward Pass.

Need for Regularisation

- Just like standard Newton, Hessians of V(x) or Q(x,u) can become indefinite in the backward pass
- Definitely necessary in the 2nd order version of DDP and often a good idea in iLQR as well (similar to Levenberg-Marquardt algorithm)
- Many options for regularising:
    - Add a multiple of identity to just like standard Newton
    - Regularize Q_{uu} as needed in the backward pass (as this is the only matrix we need to invert)
- The complexity is determined by how efficiently we compute the inversion of

  Q_{uu} (often cubic)

Improving Globalisation

Generally Armijo line search is the preferred method.
But depending on the problem in hand one can opt for other options.

Stochastic Optimal Control

Stochastic Optimal Control Problem

$$\min_u \; E[J(x,u)]$$

In comparison to the deterministic case the states are probabilistic

<u>Deterministic</u> | <u>Stochastic</u>

$$x \longrightarrow \underbrace{p(x \mid y)}_{\text{PDF of the state conditioned on measurements}}$$

DP Recursion

Assumptions

- Linear dynamics and quadratic cost
- The Noise is zero mean gaussian noise.

$$V_N(x) = E[x_N^T Q_N x_N] = E[x_N^T P_N x_N]$$

$$V_{N-1}(x) = \min_u E\Big[x_{N-1}^T Q x_{N-1} + u_{N-1}^T R u_{N-1} + (Ax_{N-1} + Bu_{N-1} + w_{N-1})^T P_N (Ax_{N-1} + Bu_{N-1} + w_{N-1})\Big]$$

$$
= \min_{u} \underbrace{E\left[x_{N-1}^T Q x_{N-1} + u_{N-1}^T R u_{N-1} + (A x_{N-1} + B u_{N-1})^T P_N (A x_{N-1} + B u_{N-1})\right]}_{\text{Standard LQR}}
$$

$$
+ \underbrace{E\Big[\underbrace{(A x_{N-1} + B u_{N-1})^T P_N w_{N-1}}_{0} \quad \underbrace{w_{N-1}^T P_N (A x_{N-1} + B u_{N-1})}_{0} + \underbrace{w_{N-1}^T P_N w_{N-1}}_{\text{constant}}\Big]}
$$

So the Noise terms do not affect the controller.

Just there is an extra term in the cost.

Some Recent Works:

1) DDP for articulated soft robots (without contacts) (under review)
2) DDP for articulated soft robots with contacts (manuscript under preparation)

Objective

"Optimal control of Articulated Soft Robots": Under Review

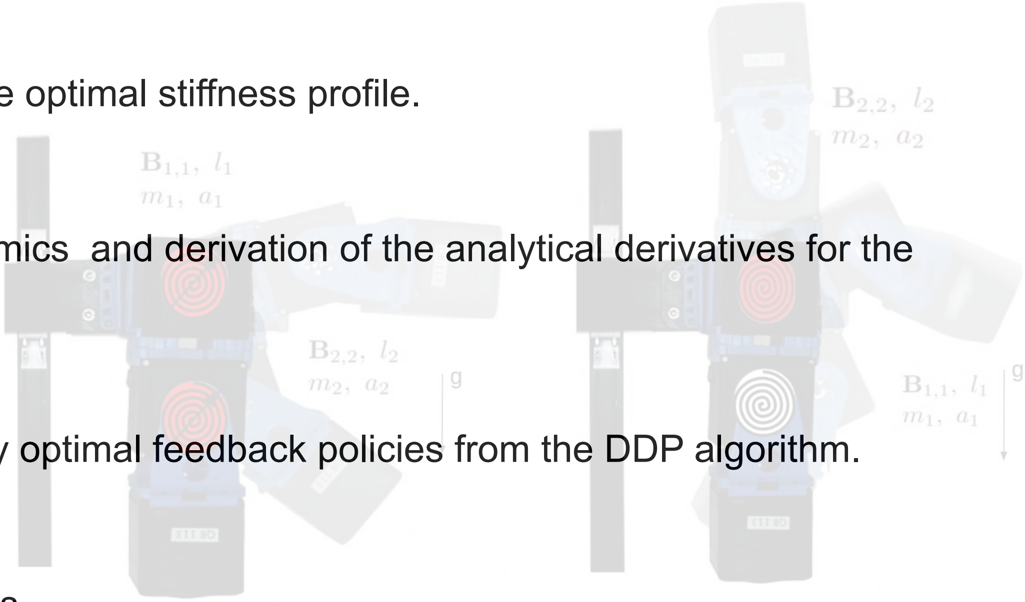
- Devise an optimal control formulation for flexible link robots and soft robots actuated by SEA/ VSA. DDP is the optimal control method used in this work.

- Plan state trajectory, input torques and the optimal stiffness profile.

- Efficient computation of the forward dynamics and derivation of the analytical derivatives for the articulated soft robot dynamic model.

- State-feedback controller based on locally optimal feedback policies from the DDP algorithm.

- Validation via simulations and experiments



Dynamic Model

- Flexible Link:

Coriolis Matrix

Gravitational Matrix

Inertia matrix

$$\mathbf{M}(\mathbf{q}(t))\ddot{\mathbf{q}}(t) + \mathbf{C}(\mathbf{q}(t), \dot{\mathbf{q}}(t))\dot{\mathbf{q}}(t) + \mathbf{G}(\mathbf{q}(t)) + \mathbf{K}(t)(\mathbf{q}(t) - \mathbf{S}\boldsymbol{\theta}(t)) = \mathbf{0},$$
$$\mathbf{B}\ddot{\boldsymbol{\theta}}(t) + \mathbf{S}^{\top}\mathbf{K}(t)(\mathbf{S}\boldsymbol{\theta}(t) - \mathbf{q}(t)) - \boldsymbol{\tau}(t) = \mathbf{0}.$$

Motor Inertia matrix

Selection matrix

Stiffness matrix

External Torque

- SEA / VSA:

$$\mathbf{M}(\mathbf{q}(t))\ddot{\mathbf{q}}(t) + \mathbf{C}(\mathbf{q}(t), \dot{\mathbf{q}}(t))\dot{\mathbf{q}}(t) + \mathbf{G}(\mathbf{q}(t)) + \mathbf{K}(t)\mathbf{q}(t) - \boldsymbol{\theta}(t)) = \mathbf{0},$$
$$\mathbf{B}\ddot{\boldsymbol{\theta}}(t) + \mathbf{K}(t)(\boldsymbol{\theta}(t) - \mathbf{q}(t)) - \boldsymbol{\tau} = \mathbf{0},$$

Optimal Control Problem

$$\min_{(\mathbf{q}_s,\dot{\mathbf{q}}_s,\boldsymbol{\theta}_s,\dot{\boldsymbol{\theta}}_s),(\boldsymbol{\tau}_s)} \ell(\mathbf{q}_N,\dot{\mathbf{q}}_N,\boldsymbol{\theta}_N,\dot{\boldsymbol{\theta}}_N)$$

$$+\sum_{k=0}^{N-1}\int_{t_k}^{t_{k+1}} \ell(\mathbf{q}_k,\dot{\mathbf{q}}_k,\boldsymbol{\theta}_k,\dot{\boldsymbol{\theta}}_k,\boldsymbol{\tau}_k)dt \quad ,$$

Objective function

$$\text{s.t.}\quad [\mathbf{q}_{k+1},\dot{\mathbf{q}}_{k+1},\boldsymbol{\theta}_{k+1},\dot{\boldsymbol{\theta}}_{k+1}] = \psi(\mathbf{q}_k,\dot{\mathbf{q}}_k,\ddot{\mathbf{q}},\boldsymbol{\theta}_k,\dot{\boldsymbol{\theta}}_k),$$

Numerical Integrator

$$[\ddot{\mathbf{q}}_k,\ddot{\boldsymbol{\theta}}_k] = \mathrm{FD}(\mathbf{q}_k,\dot{\mathbf{q}}_k,\boldsymbol{\theta}_k,\dot{\boldsymbol{\theta}}_k,\boldsymbol{\tau}_k),$$

Forward Dynamics

$$[\mathbf{q}_k,\boldsymbol{\theta}_k]\in\mathcal{Q},[\dot{\mathbf{q}}_k,\dot{\boldsymbol{\theta}}_k]\in\mathcal{V},\boldsymbol{\tau}_k\in\mathcal{U},$$

Constraint Set

Background on DDP

Forward Pass

$$\hat{\mathbf{u}}_\mathbf{k} = \mathbf{u}_\mathbf{k} + \alpha\hat{\mathbf{k}} + \hat{\mathbf{K}}(\hat{\mathbf{x}}_\mathbf{k} - \mathbf{x}_\mathbf{k})$$
$$\hat{\mathbf{x}}_\mathbf{k+1} = \mathbf{f}_\mathbf{k}(\hat{\mathbf{x}}_\mathbf{k},\hat{\mathbf{u}}_\mathbf{k})$$

Backward Pass

$$\delta\mathbf{u} = \arg\min_{\delta\mathbf{u}} \mathbf{Q}(\delta\mathbf{x},\delta\mathbf{u}) = -\hat{\mathbf{k}} - \hat{\mathbf{K}}\delta\mathbf{x}$$

$$\hat{\mathbf{k}} = \mathbf{Q}_\mathbf{uu}^{-1}\mathbf{Q}_\mathbf{u} \qquad \hat{\mathbf{K}} = \mathbf{Q}_\mathbf{uu}^{-1}\mathbf{Q}_\mathbf{ux}$$

Dynamics Computation

Model

$$\begin{bmatrix} \ddot{\mathbf{q}} \\ \ddot{\boldsymbol{\theta}} \end{bmatrix} = \begin{bmatrix} \mathrm{M} & 0 \\ 0 & B \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{\tau}_l \\ \boldsymbol{\tau}_m \end{bmatrix}$$

Analytical Derivatives

$$\begin{bmatrix} \delta\ddot{\mathbf{q}} \\ \delta\ddot{\boldsymbol{\theta}} \end{bmatrix} = - \begin{bmatrix} \mathrm{M} & 0 \\ 0 & B \end{bmatrix}^{-1} \left(\begin{bmatrix} \frac{\partial \boldsymbol{\tau}_l}{\partial \boldsymbol{x}} \\ \frac{\partial \boldsymbol{\tau}_m}{\partial \boldsymbol{x}} \end{bmatrix} \delta x + \begin{bmatrix} \frac{\partial \boldsymbol{\tau}_l}{\partial \boldsymbol{u}} \\ \frac{\partial \boldsymbol{\tau}_m}{\partial \boldsymbol{u}} \end{bmatrix} \delta u \right)$$

where,

$$\boldsymbol{\tau}_l \triangleq -\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{G}(\mathbf{q}) - \mathbf{K}(\mathbf{q} - \boldsymbol{\theta}),$$
$$\boldsymbol{\tau}_m \triangleq \mathbf{K}(\boldsymbol{\theta} - \mathbf{q}) + \boldsymbol{\tau},$$

- The effective inertia matrix is block diagonal
- Forward dynamics can be efficiently computed by using articulated body algorithm for the rigid side dynamics. The motor side inertia can be analytically inverted and the dynamics can be computed using this.

Analytical Derivatives Computation

SEA :

$$\frac{\partial \boldsymbol{\tau}_l}{\partial \mathbf{q}} = -\frac{\partial C(q,\dot{q})}{\partial \mathbf{q}} - \frac{\partial \mathbf{G}(\mathbf{q})}{\partial \mathbf{q}} - \mathbf{K},$$

$$\frac{\partial \boldsymbol{\tau}_l}{\partial \dot{\mathbf{q}}} = -\frac{\partial C(\mathbf{q},\dot{\mathbf{q}})}{\partial \dot{\mathbf{q}}}, \qquad \frac{\partial \boldsymbol{\tau}_l}{\partial \boldsymbol{\theta}} = -\mathbf{K},$$

$$\frac{\partial \boldsymbol{\tau}_m}{\partial \mathbf{q}} = \mathbf{K} + \frac{\partial \boldsymbol{\tau}}{\partial \mathbf{q}}, \qquad \frac{\partial \boldsymbol{\tau}_m}{\partial \boldsymbol{\theta}} = \boldsymbol{K},$$

VSA:

$$\frac{\partial \boldsymbol{\tau}_m}{\partial \boldsymbol{\tau}} = \mathbf{I}, \qquad \frac{\partial \boldsymbol{\tau}_m}{\partial \boldsymbol{\sigma}} = \boldsymbol{\theta} - \mathbf{q}, \qquad \frac{\partial \boldsymbol{\tau}_l}{\partial \boldsymbol{\sigma}} = \mathbf{q} - \boldsymbol{\theta}$$

\+ terms same as SEA case

Flexible Link:

$$\frac{\partial \boldsymbol{\tau}_m}{\partial \boldsymbol{\theta}} = -\mathbf{S}^\top \mathbf{K}\mathbf{S}, \qquad \frac{\partial \boldsymbol{\tau}_m}{\partial \mathbf{q}} = \mathbf{S}^\top \mathbf{K}, \qquad \frac{\partial \boldsymbol{\tau}_m}{\partial \boldsymbol{\sigma}} = \mathbf{S}^\top (\mathbf{S}\boldsymbol{\theta} - \mathbf{q}), \qquad \frac{\partial \boldsymbol{\tau}_l}{\partial \boldsymbol{\theta}} = \mathbf{K}\mathbf{S}, \qquad \frac{\partial \boldsymbol{\tau}_l}{\partial \boldsymbol{\sigma}} = -(\mathbf{q} - \mathbf{S}\boldsymbol{\theta})$$

\+ terms same as SEA case

State-Feedback Controller

We use the locally optimal policy from the backward pass of DDP in the state-feedback controller.

Feedforward term

Feedback Gain

$$\mathbf{u} = \overbrace{\mathbf{Q}_{uu}^{-1}\mathbf{Q}_u}^{\hat{\mathbf{k}}} + \overbrace{\mathbf{Q}_{uu}^{-1}\mathbf{Q}_{ux}}^{\hat{\mathbf{K}}}(\mathbf{x} - \mathbf{x}^*)$$

Simulation Validation

2 DoF SEA:
End-effector
regulation task

2 DoF SEA:
End-effector
regulation task

Flexible link
first joint is
actuated with
SEA:
Swing Up task

Flexible link
First joint is
actuated with
VSA:
Swing Up task

7 DoF SEA:
End-effector
regulation task

7 DoF VSA
End-effector
regulation task

"Optimal control of Articulated Soft Robots": Under Review

Energy Efficiency

TABLE III

POWER CONSUMPTION COMPARISON BETWEEN RIGID AND SOFT ACTUATORS

Problems	rigid	SEA	VSA
2DoF	142.07	138.46	84.17
2DoF Flexible	101.019	87.53	50.84
7DoF	> 10000	6112.77	4545.26

Experimental Validation

Control Inputs

θ_e Equilibrium position
θ_s Desired Stiffness

SEA

θ_e Optimal policy
θ_s Constant

VSA

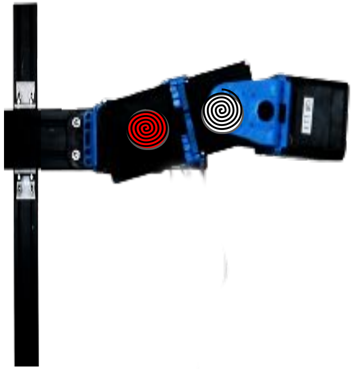
θ_e Optimal policy
θ_s Optimal policy

θ_e Set Null
θ_s Constant

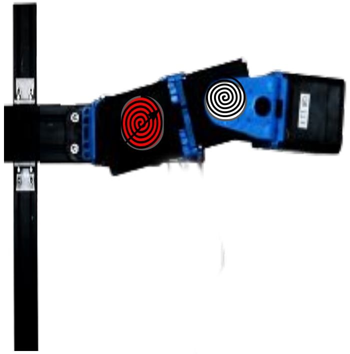
Unactuated Constant stiffness element

2DoF SEA at each joint

2DoF VSA at each joint



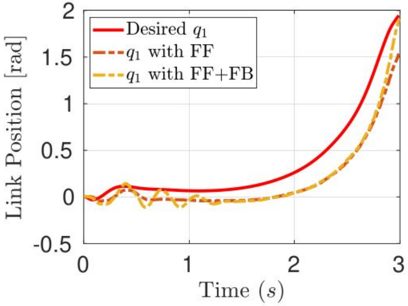
Flexible link with SEA in first joint



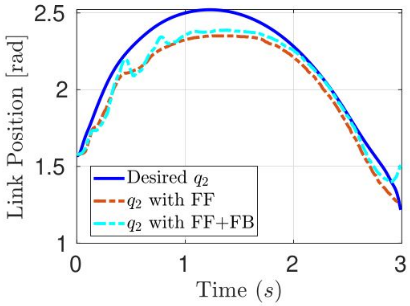
Flexible link with VSA in first joint

Experimental Results

End Effector Regulation Task



Link 1 position



Link 2 position

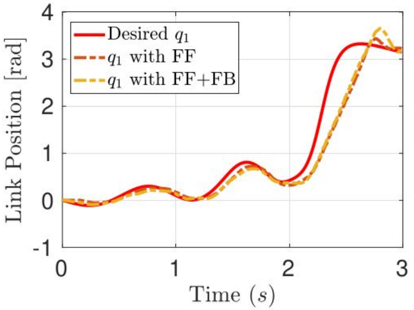
	FF	FF+FB
Joint 1	0.2503	0.2296
Joint 2	0.1274	0.1076

RMS error

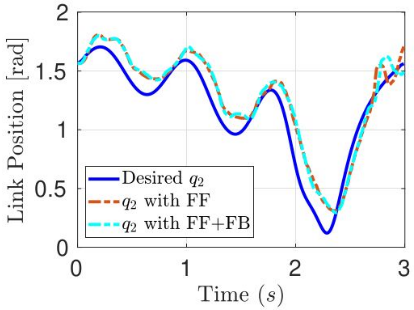
Effect of Feedback: It improves performance

Experimental Results

Flexible link with SEA in first joint



Link 1 position.



Link 2 position.

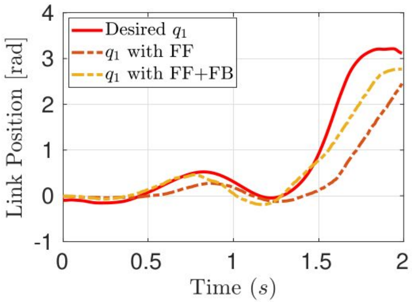
	FF	FF+FB
Joint 1	0.3908	0.3734
Joint 2	0.1607	0.1571

RMS error

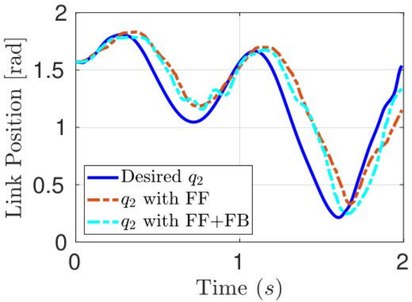
Effect of Feedback: It improves Stabilization

Experimental Results

Swing-Up with Flexible Link with VSA in first joint



Link 1 positions



Link 2 positions

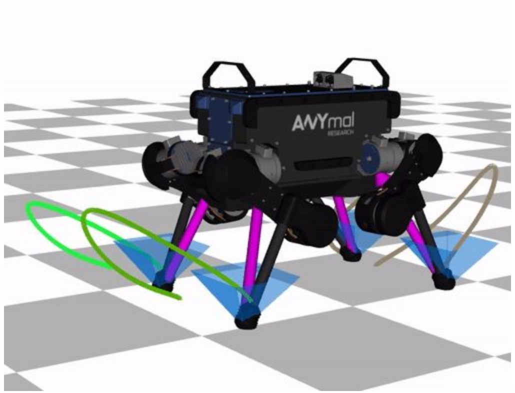
	FF	FF+FB
Joint 1	0.3857	0.2068
Joint 2	0.1701	0.1341

RMS error

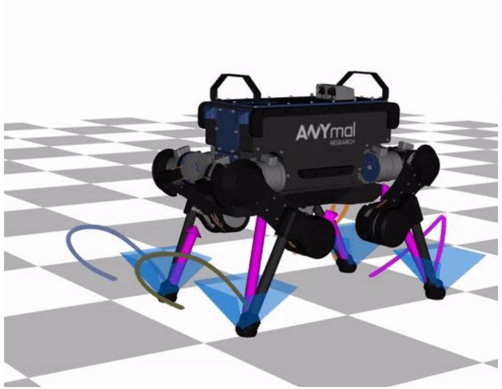
Effect of Feedback: It improves Stabilization

Results with Soft Legged Robots

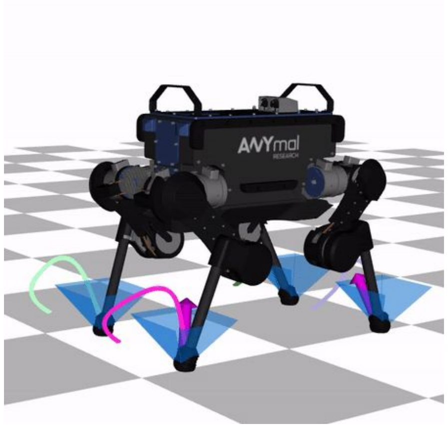
Stiffness = 5 Nm/rad



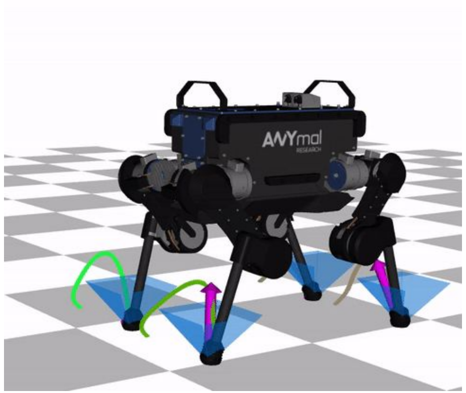
Stiffness = 10 Nm/rad

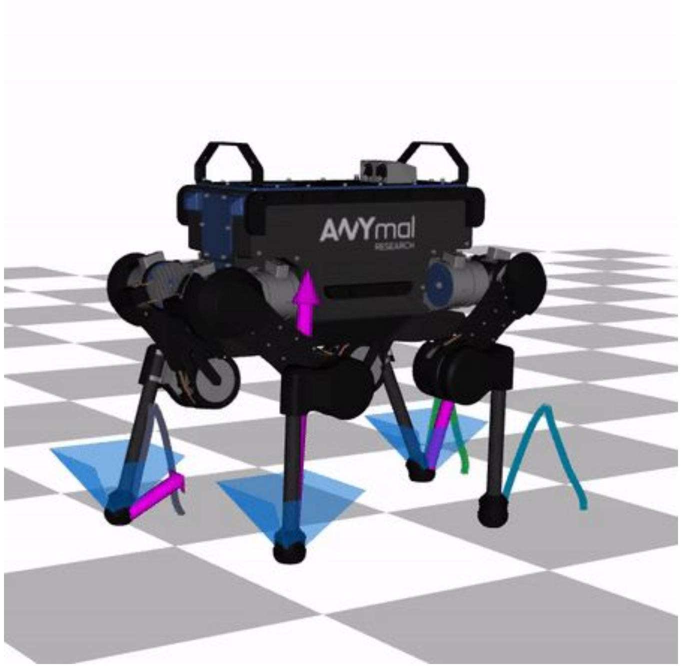


Stiffness = 30 Nm/rad



Stiffness = 100 Nm/rad





Thanks You for your patient hearing!!