16-745 Optimal Control Lecture 12

Reid Graves

February 20, 2025

1 Last Time

- Nonlinear Trajectory Optimization
- DDP/iLQR

2 Today

- DDP details + extensions
- Constraints

3 DDP recap

- Solve the unconstrained trajectory optimization problem:

$$\min_{x_{1:N}, u_{1:N-1}} \quad J = \sum_{k=1}^{N-1} l(x_k, u_k) + l_N(x_N)$$
$$s.t. \quad x_{k+1} = f(x_k, u_k)$$

- Backward Pass: - taylor expand

$$V_k(x + \Delta x) \approx V(x) + p_k^T \Delta x + \frac{1}{2} \Delta x^T P_k \Delta x$$
$$P_N = \nabla^2 l_n(x), \quad p_N = \nabla l_N(x)$$

Go backwards with Bellman backup:

$$V_{N-1}(x + \Delta x) = \min_{\Delta u} S(x + \Delta x, u + \Delta u)$$
$$\Rightarrow \Delta u_{k-1} = -d_{k-1} - K_{n-1} \Delta x_{k-1}$$
$$P_{k-1} = G_{xx} + K^T G_{uu} K - G_{xu} K - K^T G_{ux}$$
$$p_{k-1} = g_k - K^T g_u + K^T G_{uu} d - G_{xu} D$$

- Forward Rollout

$$
\begin{aligned}
&\Delta J = 0 \\
&x'_k = x_1 \\
&\quad for \quad k = 1 : N-1 : \\
&\qquad u'_k = u - \alpha d_k - K_k(x'_k - k_k) \\
&\qquad x'_{k+1} = f(x'_k, u'_k) \\
&\qquad \Delta J \leftarrow \Delta J + \alpha g_{ux} d_k \\
&end
\end{aligned}
$$

- Line search:

$$
\begin{aligned}
&\alpha = 1 \\
&do : \\
&\quad x', u', \Delta J = rollout(x, u, d, K, \alpha) \\
&\quad \alpha \leftarrow c\alpha \\
&while \quad J(x', u') < J(x, u) - b\Delta J \\
&x, u \leftarrow x', u'
\end{aligned}
$$

3.1 Examples

- Cartpole + acrobot swing up
- DDP can converge in fewer iterations but iLQR often wins in wall-clock time
- Problems are conconvex $\Rightarrow$ can land in different local optima depending on initial guess

3.2 Regularization

- Just like standard Newton, $V(x)$ and/or $S(x,u)$ Hessians can become indefinite in backward pass
- Regularization is deffinitely necessary for DDP, often a good idea with iLQR as well.
- Many options for regularizing:
  - Add a multiple of identity to $\nabla^2 S(x,u)$
  - Regularize P_k or G_k as needed in the backward pass
  - Regularize just $G_{uu} = \nabla^2_{uu} S(x,u)$ (this is the only matrix you have to invert):

$$d = G_{uu}^{-1} g_u, \quad K = G_{uu}^{-1} G_{ux}$$

- This last one is good fo iLQR but not DDP
- Regularization should not be required for iLQR but can be necessary due to floating point error.

3.3 DDP notes

3.3.1 Pros

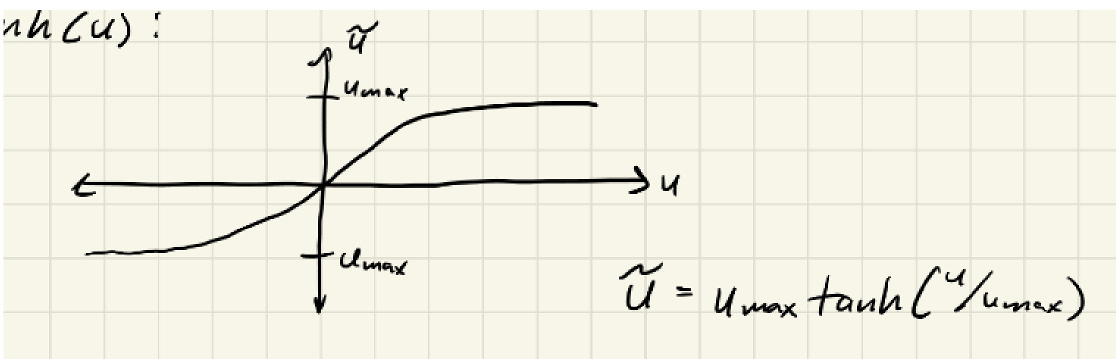
- Can be very fast (iterations + wall-clock)
- One of the most efficient methods due to exploitation of DP structure
- Always dynamically feasible due to forward rollout ⇒ can always execute on robot
- Comes with TVLQR tracking controller for free ⇒

3.3.2 Cons

- Does not natively handle constraints
- Does not support infeasible initial guess for state trajectory due to forward rollout. Bad for "maze" or "bug-trap" problems
- Can suffer from numerical ill-conditioning on long trajectories

4 Handling Constraints in DDP

- Many options depending on type of constraint
- Torque limits con be handled with a "squashing function" e.g. tanh :



- Effective, but adds nonlinearity and may need more iterations
- Better option: solve box-constrained QP in the backward pass:

$$\Delta u = \arg\min_{\Delta u} S(x + \Delta x, u + \Delta u)$$
$$s.t. \quad u_{min} \leq u + \Delta u \leq u_{max}$$

- State constraints are harder. Often penalties are added to cost function. Can cause ill-conditioning
- Better Option: Wrap entire DDP algorithm in an Augmented Lagrangian method
- Augmented Lagrangian method adds linear (multiplier) and quadratic (penalty) terms to the cost ⇒ fits into DDP nicely