RL Warm-Start for Trajectory Optimization and Control: Decision Transformer + SCP vs. IQL-LSTM + MPC/SCP

February 18, 2026

1 Problem Setting and Notation

We consider constrained trajectory generation for a dynamical system (e.g., a car). Let $x_t \in \mathcal{X} \subseteq \mathbb{R}^n$ be the state and $u_t \in \mathcal{U} \subseteq \mathbb{R}^m$ be the control at time t. Dynamics (discrete time) are

$$x_{t+1} = f(x_t, u_t), \quad t = 0, \dots, T-1. \tag{1}$$

Typical constraints include bounds, obstacle avoidance, road/lane boundaries, and actuator limits:

$$g_t(x_t, u_t) \le 0, \qquad h_t(x_t, u_t) = 0. \tag{2}$$

A nominal finite-horizon optimal control problem is

$$\min_{\{x_t, u_t\}_{t=0}^{T}} \sum_{t=0}^{T-1} \ell(x_t, u_t) + \ell_T(x_T) \quad \text{s.t.} \quad x_{t+1} = f(x_t, u_t),\ g_t \le 0,\ h_t = 0. \tag{3}$$

The goal of this document is to describe three pipelines:

1. **Decision Transformer (DT) warm-start $\rightarrow$ Sequential Convex Programming (SCP) refinement.**
2. **Implicit Q-Learning (IQL) with an LSTM policy $\rightarrow$ MPC (receding-horizon) execution.**
3. **IQL-LSTM rollout warm-start $\rightarrow$ SCP refinement** (and why it is often less natural than DT+SCP).

We then compare these approaches and recommend a path to implement.

2 Sequential Convex Programming (SCP) as the Refinement Backbone

2.1 What SCP does

SCP is a *trajectory optimization* method for nonconvex optimal control. It iteratively solves a sequence of convex approximations obtained by linearizing (or otherwise convexifying) dynamics and constraints around a reference trajectory.

Let a current reference trajectory be $\{\bar{x}_t, \bar{u}_t\}$. Linearize dynamics:

$$x_{t+1} \approx \bar{x}_{t+1} + A_t(x_t - \bar{x}_t) + B_t(u_t - \bar{u}_t), \quad A_t = \frac{\partial f}{\partial x}\Big|_{(\bar{x}_t, \bar{u}_t)}, \; B_t = \frac{\partial f}{\partial u}\Big|_{(\bar{x}_t, \bar{u}_t)}. \tag{4}$$

Similarly convexify constraints (e.g., linearization, convex inner approximation) and use a trust region:

$$\|x_t - \bar{x}_t\| \le \Delta_x, \quad \|u_t - \bar{u}_t\| \le \Delta_u. \tag{5}$$

Solve the resulting convex subproblem (QP/SOCP) to get an update, then repeat.

2.2 Why warm-start matters for SCP

SCP converges fastest (and most reliably) when the initial trajectory is:

- **close to feasible** (small constraint violations),
- **in the right local basin** (linearizations are valid),
- **geometrically consistent** over the horizon (obstacle clearance, curvature, timing).

A poor initialization can lead to many iterations, tiny trust regions, or divergence. This is exactly where learned warm-starts help.

3 Approach A: Decision Transformer Warm-Start + SCP Refinement

3.1 High-level pipeline

1. Generate an offline dataset of trajectories $\mathcal{D}$ (from an optimizer, expert, simulator, or a mix).
2. Train a sequence model (Decision Transformer or ART-style variant) to generate a *full-horizon* control/state trajectory conditioned on context.
3. At test time, use the model output as $(x^{(0)}_{0:T}, u^{(0)}_{0:T-1})$ to warm-start SCP.
4. Run SCP for a small number of iterations to obtain a feasible, locally optimal trajectory.

3.2 Decision Transformer as sequence modeling

A trajectory becomes a sequence of tokens. A common ordering is

$$(R_t, x_t, u_t)_{t=0}^{T-1}, \quad \text{where } R_t = \sum_{k=t}^{T-1} \gamma^{k-t} r_k \text{ (return-to-go)}. \tag{6}$$

DT models the conditional distribution of actions given past tokens and a desired return:

$$p_\theta(u_t \mid x_{\le t}, u_{<t}, R_{\le t}, \text{context}). \tag{7}$$

In constrained trajectory optimization, a key practical improvement (used in ART-like designs) is to also condition on a *constraint-to-go* signal, e.g.,

$$C_t = \sum_{k=t}^{T-1} c(x_k, u_k), \tag{8}$$

where $c(\cdot)$ penalizes predicted violations. Then the model can be steered toward trajectories that are easier to "repair" by SCP.

3.3 Training objective (supervised)

DT is trained as supervised learning on offline data:

$$\min_{\theta} \mathbb{E}_{(R,x,u)\sim\mathcal{D}}\Big[\sum_{t=0}^{T-1} \|u_t - \hat{u}_t\|^2\Big], \quad \hat{u}_t \sim p_\theta(\cdot \mid \text{past tokens}). \tag{9}$$

No Bellman backups; no policy gradient. The *RL* aspect is that the dataset arises from an MDP and the model is conditioned on returns/constraint budgets to produce high-performing behavior.

3.4 Warm-starting SCP with DT

At inference, DT generates either:

- a control sequence $u_{0:T-1}^{(0)}$, and then we forward simulate $x^{(0)}$ using the dynamics, or
- both $(x_{0:T}^{(0)}, u_{0:T-1}^{(0)})$ directly (often helpful because it provides a coherent state path for linearization).

Then SCP runs:

$$(\bar{x}, \bar{u}) \leftarrow (x^{(0)}, u^{(0)}) \quad \rightarrow \quad \text{convexify} \quad \rightarrow \quad \text{solve} \quad \rightarrow \quad \text{update.}$$

A strong warm start typically reduces:

- the number of SCP iterations,
- the amount of trust-region shrinking,
- the frequency of infeasible subproblems.

3.5 Key strengths of DT+SCP

- **Naturally produces full-horizon structure.** DT learns a distribution over entire trajectories, which is exactly what SCP linearizes around.
- **Stable training.** Pure supervised learning on offline trajectories.
- **Conditioning knobs.** Return-to-go and constraint-to-go provide explicit tradeoff control between performance and feasibility.
- **Multiple modes.** The model can represent distinct maneuver modes (e.g., different ways around an obstacle) better than a single local policy.

3.6 Failure modes / risks

- **Out-of-distribution (OOD) contexts.** DT can produce implausible trajectories if asked to extrapolate.
- **Dataset quality.** If the offline trajectories are low quality or inconsistent, the warm starts degrade.
- **No hard guarantees.** Feasibility is still guaranteed by SCP, not by DT.

4 Approach B: IQL-LSTM + MPC (Receding Horizon)

4.1 High-level pipeline

1. Collect an offline dataset $\mathcal{D}$ of transitions or trajectories: (x_t, u_t, r_t, x_{t+1}).
2. Train an IQL agent: a value model and a policy (here, the policy is an LSTM).
3. Use the learned policy online either:
   - as the primary controller, or
   - as a **warm-start generator** for MPC by rolling it out for N steps to propose $u^{(0)}_{0:N-1}$.

4.2 IQL in one page: what it learns

IQL is an *offline RL* method that avoids explicit maximization over actions (which can be unstable offline). It learns:

- a Q-function $Q_\phi(x, u)$,
- a value function $V_\psi(x)$ fit by expectile regression,
- a policy $\pi_\theta(u \mid x)$ trained by advantage-weighted regression.

A typical IQL structure:

$$V_\psi(x) \approx \arg\min_V \; \mathbb{E}_{(x,u)\sim\mathcal{D}}\Big[\rho_\tau\big(Q_\phi(x,u) - V\big)\Big], \quad \rho_\tau(\delta) = |\tau - \mathbf{1}\{\delta < 0\}|\delta^2, \tag{10}$$

where $\tau \in (0, 1)$ sets the expectile. Then Q_ϕ is trained with a TD-style target using V_ψ:

$$\min_\phi \; \mathbb{E}_{(x,u,r,x')\sim\mathcal{D}}\Big[\big(Q_\phi(x,u) - (r + \gamma V_\psi(x'))\big)^2\Big]. \tag{11}$$

Finally, the policy is trained to upweight actions with high estimated advantage:

$$A(x,u) = Q_\phi(x,u) - V_\psi(x), \qquad \max_\theta \; \mathbb{E}_{(x,u)\sim\mathcal{D}}\Big[\exp(\beta A(x,u)) \log \pi_\theta(u \mid x)\Big]. \tag{12}$$

4.3 Why an LSTM policy?

An LSTM policy $\pi_\theta(u_t \mid h_t)$ with hidden state h_t can handle partial observability and temporal dependencies:

$$h_t = \mathrm{LSTM}(h_{t-1}, o_t), \qquad u_t \sim \pi_\theta(\cdot \mid h_t), \tag{13}$$

where o_t are observations (possibly $o_t = x_t$ if fully observed).

4.4 MPC execution with IQL warm-start

MPC solves a horizon-N optimization at each real-time step t:

$$\min_{u_{t:t+N-1}} \sum_{k=0}^{N-1} \ell(x_{t+k}, u_{t+k}) + \ell_f(x_{t+N}) \quad \text{s.t.} \quad x_{t+k+1} = f(x_{t+k}, u_{t+k}), \text{ constraints.} \tag{14}$$

Warm-start options:

- **Shifted warm-start:** use previous MPC solution shifted by one step.
- **IQL rollout warm-start:** roll out the LSTM policy from the current state for N steps to obtain $u^{(0)}_{t:t+N-1}$, then initialize the MPC solver with it.

This is *very natural* because IQL directly provides a closed-loop policy that can propose plausible near-term controls quickly.

4.5 Key strengths of IQL-LSTM+MPC

- **Naturally aligned with feedback control.** IQL learns a policy; MPC needs a good initial control sequence *at each time step.*
- **Robustness via receding horizon.** MPC re-optimizes as the system evolves, correcting modeling errors.
- **Offline improvement potential.** Unlike pure imitation, IQL can improve over the dataset under certain conditions (in practice: depends on coverage).
- **Ensembles help.** Multiple LSTMs can provide diverse warm starts and uncertainty estimates.

4.6 Failure modes / risks

- **Offline RL brittleness.** Performance depends strongly on dataset coverage; OOD states can lead to poor proposals.
- **Constraint handling is indirect.** IQL policy is not guaranteed to respect constraints; MPC must enforce them.
- **Horizon coherence is not the first-class object.** Rolling out a policy can accumulate errors and produce trajectories that drift or oscillate over long horizons.

5 Approach C: IQL-LSTM Rollout Warm-Start + SCP Refinement

5.1 How it works

This approach is valid:

1. From current context (initial state, goal), roll out the IQL-LSTM policy for T steps to produce $u^{(0)}_{0:T-1}$.
2. Forward simulate to produce $x^{(0)}_{0:T}$.
3. Initialize SCP at $(x^{(0)}, u^{(0)})$ and refine.

5.2 Why it is often less natural than DT+SCP

Even though it works, it is often a weaker match to SCP for *trajectory generation* because:

(1) Object mismatch. SCP is a batch trajectory optimizer; it benefits from an initialization that captures full-horizon structure. An IQL policy is trained to choose actions locally to maximize value, not explicitly to produce globally consistent trajectories.

(2) Compounding rollout error. Small policy errors can compound across T steps. The resulting trajectory can be geometrically inconsistent (e.g., late obstacle avoidance), making SCP linearizations harder.

(3) Conditioning and multimodality. DT/ART-style models can condition directly on desired return and constraint budgets and can represent multiple maneuver modes. An IQL policy tends to represent a single mode unless explicitly diversified (ensembles help, but it is not the same as modeling a trajectory distribution).

(4) Practical data generation. If you already have an optimizer producing trajectories (common in planning), it is often simpler to train a sequence model to imitate those solutions than to train an offline RL agent with stable value learning.

5.3 When IQL-LSTM+SCP might be attractive

- Dataset consists of *suboptimal* behavior logs, and you want improvement beyond imitation.
- You want a *single* learned controller that can also act without SCP if needed.
- You care about disturbance rejection in the learned proposal itself (policy produces feedback behavior).

6 Comparison: DT+SCP vs IQL-LSTM+MPC vs IQL-LSTM+SCP

6.1 What each approach is "best at"

	DT + SCP	**IQL-LSTM + MPC**	**IQL-LSTM + SCP**
Primary role	Trajectory prior (full-horizon) + feasibility repair	Feedback control with constraint enforcement (online)	Policy rollout as trajectory prior + feasibility repair
Natural fit	Batch planning / trajectory optimization	Online receding-horizon control	Works, but object mismatch for long-horizon planning
Training stability	High (supervised)	Medium (offline RL can be brittle)	Medium (same as IQL)
Handles constraints	Via SCP guarantees	Via MPC guarantees	Via SCP guarantees
Multimodality	Strong (sequence distribution)	Moderate (ensembles)	Moderate (ensembles)
Best evaluation metrics	SCP iterations, feasibility rate, final cost	MPC solve time, tracking error, violations	SCP iterations + rollout coherence

6.2 Is IQL-LSTM "more natural" with MPC than SCP?

Yes, usually. IQL learns a *policy*, and MPC needs a *near-term control sequence* at every time step. Rolling out the policy for N steps to seed MPC is a direct and stable use of what the policy provides. In contrast, using a policy rollout to seed a *full-horizon batch optimizer* asks the policy to implicitly solve long-horizon geometry; it can work, but it is not what the policy is optimized for.

6.3 If your goal is trajectory generation with feasibility guarantees, is DT+SCP better than IQL-LSTM+SCP?

Typically yes. For warm-starting SCP, the most important property is a globally coherent initialization that sits near a good local basin and is easy to convexify. DT/ART-style sequence models are trained to generate coherent sequences and can be explicitly conditioned on return and constraint budgets. An IQL policy rollout can be coherent, but it is optimized indirectly through value learning and is more sensitive to dataset coverage and rollout compounding.

7 Recommended Path (for a strong project)

If the project goal is **trajectory generation under hard constraints** with a learned warm-start:

1. **Start with DT (or ART-style DT with constraint-to-go) + SCP.**
    - Generate a dataset by solving many trajectory optimization problems offline.
    - Train the sequence model to predict $(x_{0:T}, u_{0:T-1})$ (or just u plus rollout).
    - Evaluate: SCP iterations, runtime, feasibility success rate, final objective value.
2. **Add MPC as an execution layer if time permits.**
    - Use the SCP-refined plan as a reference trajectory.
    - Run tracking MPC (or NMPC) online to handle disturbances/mismatch.
3. **Use IQL-LSTM as a complementary baseline/ablation.**
    - Train IQL-LSTM and compare its rollout warm-start for SCP against DT warm-start.
    - Also compare IQL rollout warm-start for MPC against shifted warm-start.

Why this recommendation is pragmatic. DT+SCP cleanly separates responsibilities:

- Learning captures the *manifold of good trajectories* (fast generation, multimodal structure).
- SCP enforces *hard feasibility* and improves cost.
- MPC (optional) provides *closed-loop robustness* for real-time deployment.

This separation tends to minimize "RL tuning pain" while maximizing measurable gains (iteration count, solve time, feasibility).

8 Suggested Experiments and Deliverables

8.1 Baselines

Include at least:

- Straight-line or zero-control initialization $\rightarrow$ SCP

- Shifted previous-solution initialization (if applicable) $\rightarrow$ SCP/MPC
- DT warm-start $\rightarrow$ SCP
- IQL-LSTM rollout warm-start $\rightarrow$ SCP
- IQL-LSTM rollout warm-start $\rightarrow$ MPC

8.2 Metrics

- **Feasibility rate**: fraction of problems where final solution satisfies constraints.
- **Optimization effort**: SCP iterations, solver time; MPC iterations/time per step.
- **Performance**: final cost, tracking error, smoothness (comfort), clearance margins.
- **Robustness**: disturbances, friction mismatch, perception noise, goal perturbations.

8.3 A clean story to tell

A learned sequence model provides a globally coherent trajectory prior; SCP repairs feasibility and local optimality; MPC (optional) tracks robustly online. Offline RL policies are strong baselines for generating short-horizon warm-starts for MPC, but are typically less effective than DT/ART-style models for full-horizon SCP warm-starting.