Table of Contents

MPC_GPS Demystify and try to apply with unknown dynamics

I. Theory

A. Trajectory

i. DDP Trajectory Optimization

ii. Stochastic Trajectory Optimization

B. Fitting dynamics using GMM

i. Intuition

1. **Goal**: Least square regression where:
   Data: $D = \{d_n = [x_n, u_n, x'_n]\}_{n=1}^{N}$
   Hypothesis: $\hat{p}(x'|x,u) \approx \mathcal{N}\left(A\begin{bmatrix}x\\u\end{bmatrix} + b, \Sigma\right)$
   Target: x'
   Objective function: $E_D(A,b) = \frac{1}{2}\sum_{n=0}^{N}\left\{x'_n - \theta^T\begin{bmatrix}x\\u\end{bmatrix}\right\}^2 + \frac{\lambda}{2}R(\theta)$.
   where $R(\theta)$ is regularized term, and $\theta \sim p(\theta)$. In this case $p(\theta)$ is fitted by sample from current and previous iteration, and then inference from current sample (check below).

2. **Idea**: The regularized least square is also equivalent to Maximum A Posterior $p(x', \theta|x,u) \propto p(x'|x,u,\theta)p(\theta)$.
   GPS solve this by first construct a joint distribution $p(x', x, u)$, then find θ_{MAP} of the posterior $p(\theta|x,u,x') \propto p(x', x, u|\theta)p(\theta)$, conditioning it and get $p(x'|x,u)$, since the joint distribution is Gaussian.

3. **Solution**

   3.1. Update prior parameters: $\pi_k, \mu_k, \Sigma_k \forall k$ by fitting $V = \{v_n = [x_n, u_n, x'_n]\}_{n=1}^{N}$ into GMM.
   Note: N also included sample from **previous** iteration. E.g, for 4 iterations
   $N = \sum_{k=i-3}^{i} \#sample_k$.

   3.2. Now inference from GMM above to get prior distribution $p(\theta)$ on current (smaller) data .

3.3. Find θ_{MAP} of the posterior $p(\theta|x,u,x') \propto p(x',x,u|\theta)p(\theta)$, and then conditioning on this to get the final μ, Σ of $p(x'|x,u)$.

ii. Implementation

1. ***Update_prior*** with current & 3 previous iterations:

   1.1. Model the joint distribution of input $\{x_t, u_t\}$ and output x_{t+1} as a Gaussian Mixture Model (GMM). Therefore, log likelihood is

   $$ln\ p(V|\pi,\mu,\Sigma) = \sum_{n=1}^{N} ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(v_n|\mu_k,\Sigma_k) \right\}.$$

   1.2. This likelihood is maximized using EM procedure (Copy from *Bishop 2006 book's*).

   E-step:

   $$\gamma(z_k) = \frac{\pi_k N(\mathrm{x}\,|\,\mu_k,\Sigma_k)}{\sum_{j=1}^{K} \pi_j N(\mathrm{x}\,|\,\mu_j,\Sigma_j))}$$

   M-step:

   $$\mu_k^{\text{new}} = \frac{1}{N_k}\sum_{n=1}^{N} \gamma(z_{nk})\mathrm{x}_n$$

   $$\Sigma_k^{\text{new}} = \frac{1}{N_k}\sum_{n=1}^{N} \gamma(z_{nk})(\mathrm{x}_n - \mu_k^{\text{new}})(\mathrm{x}_n - \mu_k^{\text{new}})^T$$

   $$\pi_k^{\text{new}} = \frac{N_k}{N} \qquad \text{where} \qquad N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

   1.3. **Confusing**: Why in M-step they (source code) update $\Sigma_k = w_k x x^T + \mu_k \mu_k^T$, while the right one should be $\Sigma_k = w_k(x-\mu_k)(x-\mu_k)^T$ (above)?

2. ***Fit*** with sample D in current iteration:

   2.1. Likelihood is Gaussian: $p(D|\mu,\Sigma) = \mathcal{N}(\mu,\Sigma)$, where $D = \{d_n = [x_n, u_n, x'_n]\}_{n=1}^{N}$ with $N = \#sample_k$ (*current* iteration).

   2.2. Compute μ_0, Φ, m, n_0 by inference parameter base on current data (small) from the GMM (fitted on large data):

   a) Compute posterior cluster weights (for D):

   $$\pi_k = \frac{N_k}{N} = \frac{\sum_{n=1}^{N} \gamma(z_{nk})}{N}$$

   b) Moments

   - $\bar{\mu} = \sum_{k=1}^{K} \pi_k \mu_k$
   - $\bar{\Sigma} = \sum_{k=1}^{K} \pi_k \left[\Sigma_k + (\mu_k - \bar{\mu})(\mu_k - \bar{\mu})^T\right]$

c) Compute prior: $\mu_0 = \bar{\mu}, \Phi = n_0\bar{\Sigma}$.

- **Confusing**: In source code, they calculate $m = 1.0, n_0 = \frac{N-2-d}{N}$. I have no idea why they use this?

2.3. Compute joint distribution:

- Given: prior $p(\mu, \Sigma) = NIW(\mu_0, m, \Phi, n_0)$ and likelihood $p(D|\mu, \Sigma) = \mathcal{N}(\mu, \Sigma)$.
- Compute joint distribution (Bayesian Inference from Wiki): $p(D, \mu, \Sigma) = p(D|\mu, \Sigma)p(\mu, \Sigma) = NIW(\mu_k, m_k, \Phi_k, n_k)$, where
  - $\mu_k = \frac{m\mu_0 + n_0\hat{\mu}}{m + n_0}$
  - $\Phi_k = \Phi + N\hat{\Sigma} + \frac{Nm}{N+m}(\hat{\mu} - \mu_0)(\hat{\mu} - \mu_0)^T$
  - **Note**: $\hat{\mu}, \hat{\Sigma}$ are empirical mean and covariance of dataset D.
- Calculate μ_{MAP}, Σ_{MAP} of joint distribution (or posterior) by calculate the mode (point that has the maximum pdf):
  - $\mu_{MAP} = \hat{\mu}$
  - $\Sigma_{MAP} = \frac{\Phi_k}{n_k + p + 1}$, where $n_k = n_0 + N + 1$
- **Confusing**: In *End-to-End GPS from C. Finn*, they said $\Sigma_{MAP} = \frac{\Phi_k}{n_k}$ ($n_k = n_0 + N$). It seem legit because the prior parameter $\Phi = n_0\Sigma$.

2.4. Conditioning on this and get the result $\hat{p}(x_{t+1}|x_t, u_t) \approx \mathcal{N}\left(f_{x_t,u_t}\begin{bmatrix}x_t\\u_t\end{bmatrix} + b, \Sigma\right)$.

- $\mu = \Sigma_{x',[x,u]}\Sigma^{-1}_{[x,u],[x,u]}\begin{bmatrix}x_t\\u_t\end{bmatrix} + \left(\mu_{x'} - \Sigma_{x',[x,u]}\Sigma^{-1}_{[x,u],[x,u]}\mu_{[x,u]}\right)$
- $\Sigma = \Sigma_{x',x'} - \Sigma_{x',[x,u]}\Sigma^{-1}_{[x,u],[x,u]}\Sigma_{[x,u],x'}$

C. MPC with unknown dynamics

1. *Input*: offline trajectory distribution $p(u_t|x_t)$, fitted dynamics $p(x_{t+1}|x_t)$, current state x_t.
2. *Goal*: follow offline trajectory.
3. *Output*: MPC trajectory distribution $q(u_{t'}|x_t),\ \forall t' \in [t+1, t+H]$.
4. **Problem**:

4.1. MPC is online trajectory optimization with 2 main attribute:

a) Use current state x_t as a initial state → Feedback control.

b) Short horizon → Need to re-optimize many time → Online optimization.

4.2. Surrogate cost criteria from *MPC_GPS paper*:

a) Encourage visit states have high probability from offline LQG solution $p(\tau) = p(u_t|x_t) * p(x_{t+1}|x_t)$.

b) Produce good long-horizon behavior while it is short-horizon.

c) Behavior close to neural network policy $\pi_\theta(u_t|x_t)$.

5. **Mathematics solution**:

5.1. From 4.1 a) I need to have initial state is current state x_t.
Compute $p(x_{t'}|x_t)$ by forward pass **with known** x_t using $p(u_{t'}|x_{t'})$, $p(x_{t'+1}|x_{t'})$.

5.2. From 4.2 a & b → The meaningful choice (best from my knowledge) of cost is: cost low (better) when $p(x_{t'}|x_t)$ high → $-log\ p(x_{t'}|x_t)$ is chosen.

5.3. From 4.2 c → MPC solution need to close to neural network policy $\pi_\theta(u_t|x_t)$.
But for now, I just want to use MPC as a trajectory optimization, not as a demonstrator from MPC_GPS. So instead of $\pi_\theta(u_t|x_t)$, I use $p(u_t|x_t)$ - offline LQG solution.
Note: I tried to use $-log\ p(x_{t'}|x_t)$ alone, but it doesn't work.

5.4. The final surrogate cost become: $\tilde{l}(x_{t'}, u_{t'}) = -\ log\ p(x_{t'}|x_t) - log\ p(u_{t'}|x_{t'})$.
Note: I get rid of ν, and don't use η as LQG instead. Check my question 2 below for this.

5.5. The final problem is: $\min_{q_{ij}(u_t|x_t)} E_{p(x_t,u_t)}[\tilde{l}(x_t, u_t)] - \mathcal{H}(q_{ij}(u_t|x_t))$

5.6. The solution of this problem is the same as LQG,
$q_{ij}(u_t|x_t) = \mathcal{N}(\tilde{K}_{tij}x_t + \tilde{k}_{tij}, \tilde{Q}^{-1}_{u,utij})$.

a) To compute $\tilde{K}$, $\tilde{k}$, we need to compute gradient and Hessian of $\tilde{l}(x_t, u_t)$.
$\frac{\partial}{\partial x}(-\ log\ p(x_{t'}|x_t)) = \Sigma^{-1}_{t'}(x_{t'} - \mu_{t'})$. In this case $x_{t'}$ = forward pass **with unknown** x_t, and use x_0 as initial state using $p(u_{t'}|x_{t'})$, $p(x_{t'+1}|x_{t'})$.

b) $\frac{\partial^2}{\partial x^2}(-\ log\ p(x_{t'}|x_t)) = \Sigma^{-1}_{t'}$.
$\frac{\partial}{\partial x}(-\ log\ p(u_{t'}|x_t))$, $\frac{\partial^2}{\partial x^2}(-\ log\ p(u_{t'}|x_t))$ are known.

D. Improvement

i. Line search

1. Bracket Line search:

1.1. Use bracket line search for η to find the updated distribution $p(u_t|x_t)$that KL divergence satisfy the constrained $KL(p(u_t|x_t)\|\hat{p}(u_t|x_t)) \leq \epsilon$.

1.2. Assume $p(x_t) = \mathcal{N}(\mu, \Sigma)$, $p(u_t|x_t) = \mathcal{N}(\mu_{u1}, A)$, $\hat{p}(u_t|x_t) = \mathcal{N}(\mu_{u0}, \hat{A})$

1.3.

$$KL = E_{p(x_t)}\left[log\ \frac{p(u_t|x_t)}{\hat{p}(u_t|x_t)}\right]$$
$$= \frac{1}{2}E_{p(x_t)}\left[-log\,|A| + (x-\mu_{u1})^T A^{-1}(x-\mu_{u1}) + log\left|\hat{A}\right| - (x-\mu_{u0})^T \hat{A}^{-1}(x-\mu_{u0}))\right]$$

We have: $\left\{(x-\mu_u)^T A^{-1}(x-\mu_u)\right\} \approx \frac{1}{2}x^T Mx + xv + c$

Where $M = \frac{\partial^2\left\{(x-\mu_u)^T A^{-1}(x-\mu_u)\right\}}{\partial^2 x}$, $v = \frac{\partial\left\{(x-\mu_u)^T A^{-1}(x-\mu_u)\right\}}{\partial x}$

Then, $KL = \frac{1}{2}E_{p(x_t)}\left[-log\,|A| + \frac{1}{2}x^T Mx + xv + c + log\left|\hat{A}\right| - \frac{1}{2}x^T\hat{M}x - x\hat{v} - \hat{c}\right)$

$$= \left\{\frac{1}{2}\mu(M-\hat{M})\mu + \frac{1}{2}Tr(\Sigma(M-\hat{M})) + \mu(v-\hat{v}) + c - \hat{c}\right\} + \frac{1}{2}log\ \frac{\left|\hat{A}\right|}{|A|}$$

2. In MPC trajectory optimization, because of unconstrained problem, so I can't use the procedure above to make $Q_{uu} \succeq 0$ (always positive definite).

   2.1. So from *Synthesis and Stabilization of Complex Behaviors through Online Trajectory Optimization 2012*, I use the old regularization trick: $\tilde{Q}_{uu} = Q_{uu} + \eta\mathcal{I}_m$

   2.2. η is updated using bracket line search, but $\eta_0 = 0$ (original is 1, and use η from previous iteration)
   Question: Does it work if $\eta = 10^{-4}$ (small value), because to make Q_{uu} not singular matrix (full rank ??).

ii. Dual step_size of KL divergence constraint

1. From *Learning Contact-Rich Manipulation Skills with Guided Policy Search 2015*, the ϵ step size update by using model improvement.

2. Intuition: a larger cost the further we deviate from the previous policy (*from Guided Policy Search as Approximate Mirror Descent 2016*).
   The reasonable ϵ is the ϵ that maximize actual cost improvement.

   2.1. To do that, we must model the actual cost = $f(\epsilon) = a\epsilon^2 + b\epsilon$.

   $actual\_cost = predicted\_cost + noise \Rightarrow b = \frac{predicted\_cost}{\epsilon}$.

   From that can find a when b, actual cost are known.

   2.2. Final problem: $\max_{\epsilon} f(\epsilon) = a\epsilon^2 + b\epsilon \Rightarrow \epsilon = \frac{-b}{2a}$.

   2.3. To pick new ϵ', $b = \frac{predicted}{\epsilon}$ where ϵ from previous iteration.

   2.4. Note: From assumption "a larger cost the further we deviate from the previous policy":
   $KL = \frac{1}{2}\left[\log\frac{|\Sigma_2|}{|\Sigma_1|} - d + \mathrm{tr}(\Sigma_2^{-1}\Sigma_1) + (\mu_2-\mu_1)^T\Sigma_2^{-1}(\mu_2-\mu_1)\right]$ (it linear in

covariance Σ, quadratic in mean μ) .
$actual\_cost = l_k^k - l_{k-1}^{k-1}$, and its $E[actual\_cost]$ is computed below, has the same behavior (linear in covariance Σ, quadratic in mean μ).

1. To compute (Laplace approximation?? - Check Question 3 below) predicted, actual cost of $E_{p(\tau)}[l(\tau)]$.
   1.1. $l(\tau) \approx l(x,u) + \begin{bmatrix} x \\ u \end{bmatrix}^T \frac{\partial l(\tau)}{\partial xu} + \frac{1}{2} \begin{bmatrix} x \\ u \end{bmatrix}^T \frac{\partial^2 l(\tau)}{\partial xu, xu} \begin{bmatrix} x \\ u \end{bmatrix}$.
   1.2. $E[x^T A x] = Tr(A\Sigma) + \mu^T A \mu$ (From Matrix cookbook).

II. Implement

A. Design

B. Point mass

i. Add obstacle

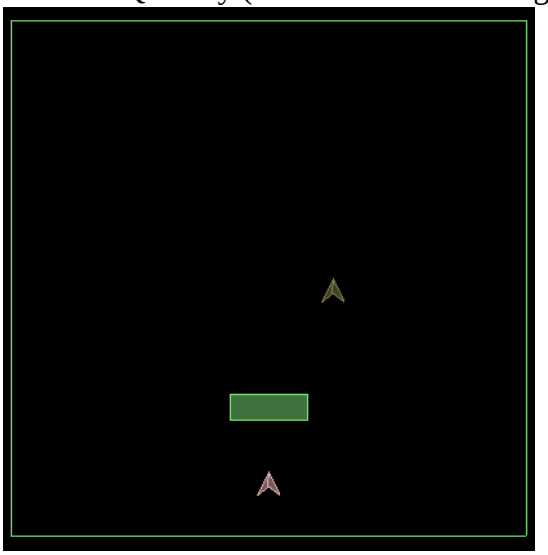
C. Arm

III. Result

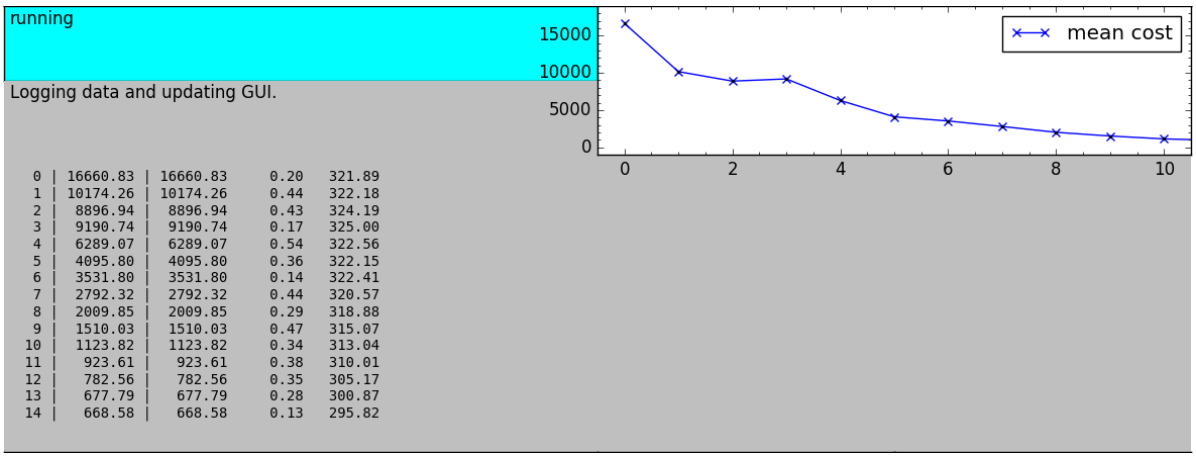
A. Box2D

i. Point mass

1. Offline LQG only (The different from original is that the obstacle)



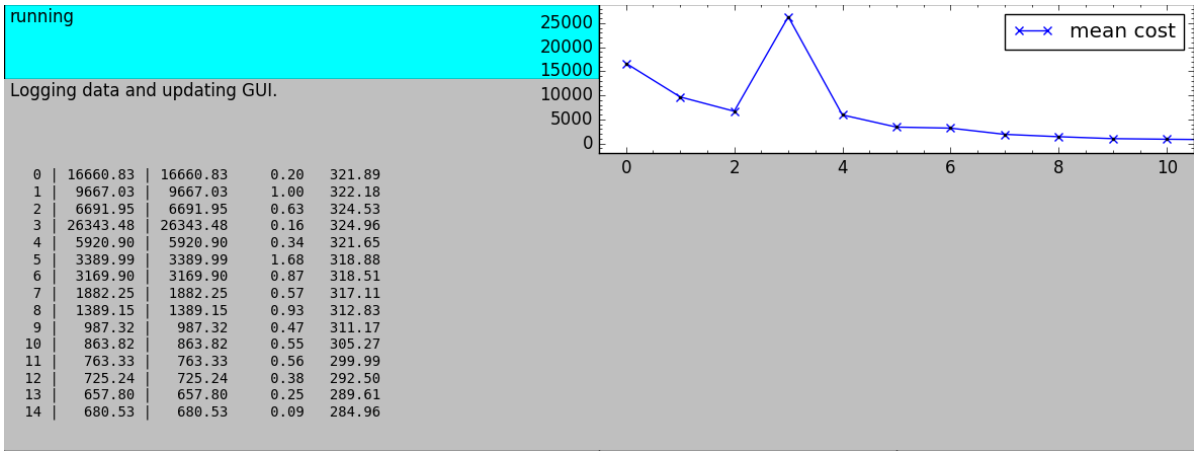
1.1. Setup: 15 iterations, 1 obstacle, 1 condition (initial state), 5 samples, T: 100



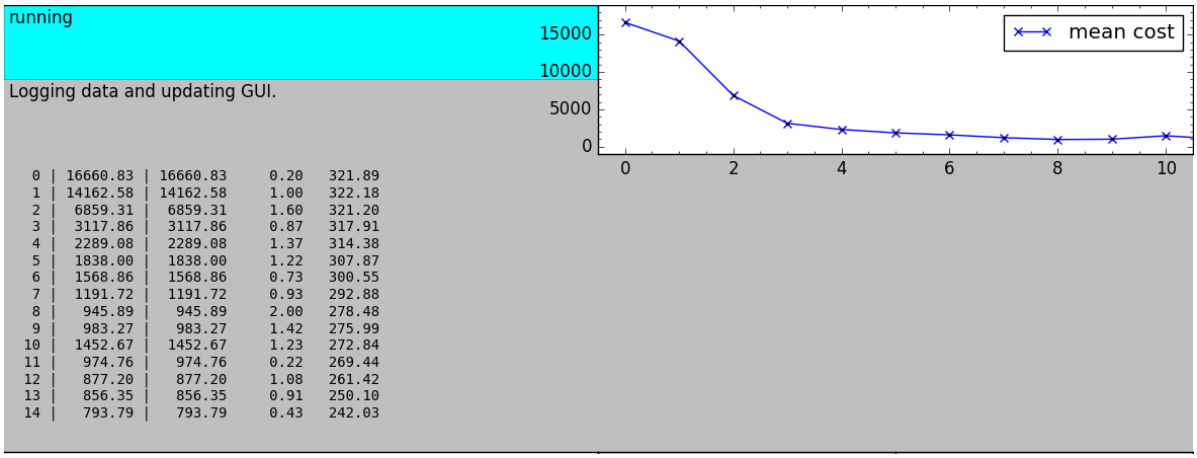
1.2. Above image is the result: Row from left to right: iteration, avg_cost, cost, step, entropy.

2. MPC unconstrained (Check Question 1 below)

2.1. Setup: 15 iterations, 1 obstacle, 1 condition (initial state), 5 samples, T: 100



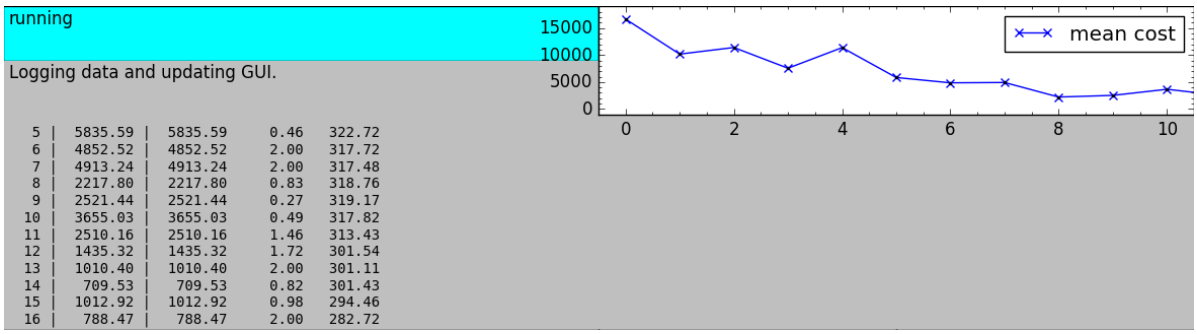
2.2. Short horizon: 10



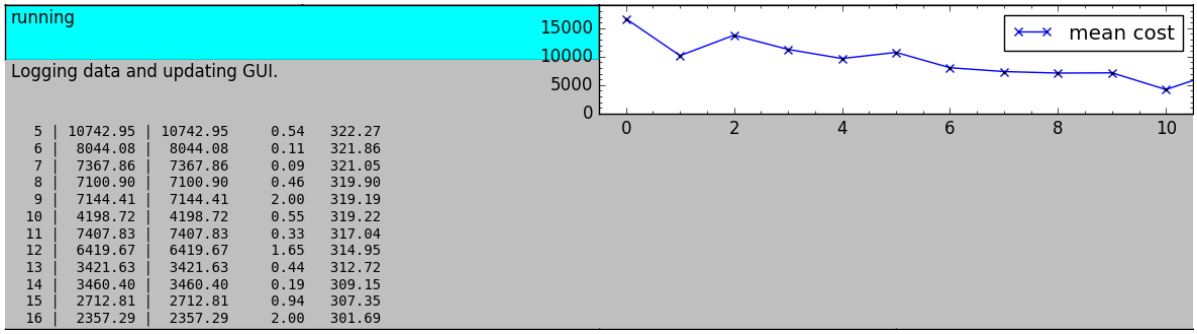
2.3. Short horizon: 5

3. MPC constrained (solved by DGD)

3.1. Setup: same as MPC unconstrained



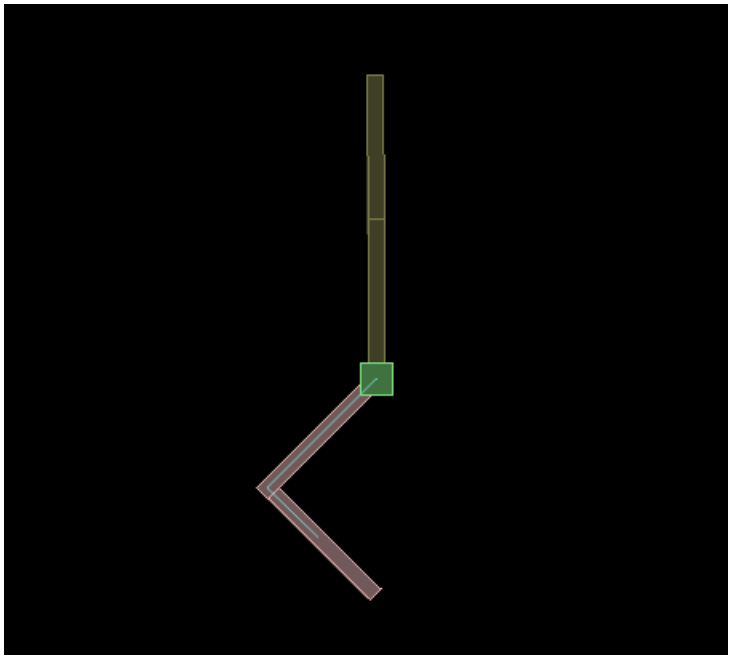
3.2. Short horizon: 10. It work but the cost quite jaggle.



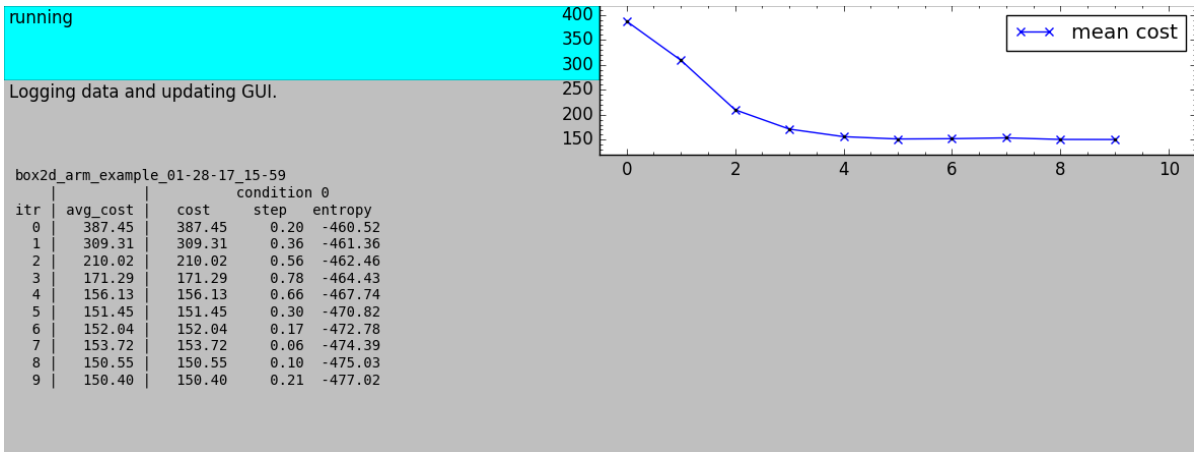
3.3. Short horizon: 5. It cost still high at iteration 16.

ii. 2D link Arm

1. Offline LQG only (exactly the same as origin Arm world).



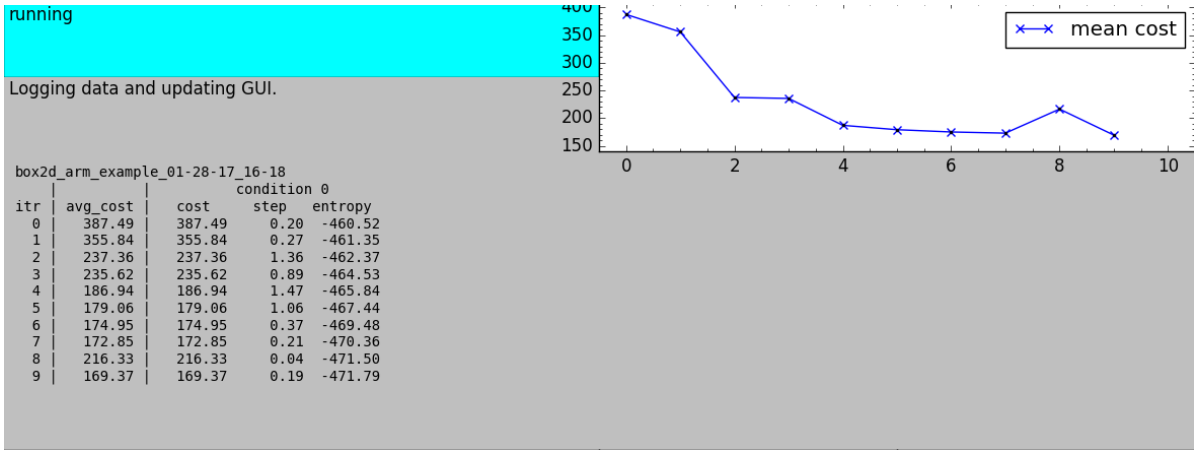
1.1. Setup: 10 iterations, 1 condition (initial state), 5 samples, T: 100



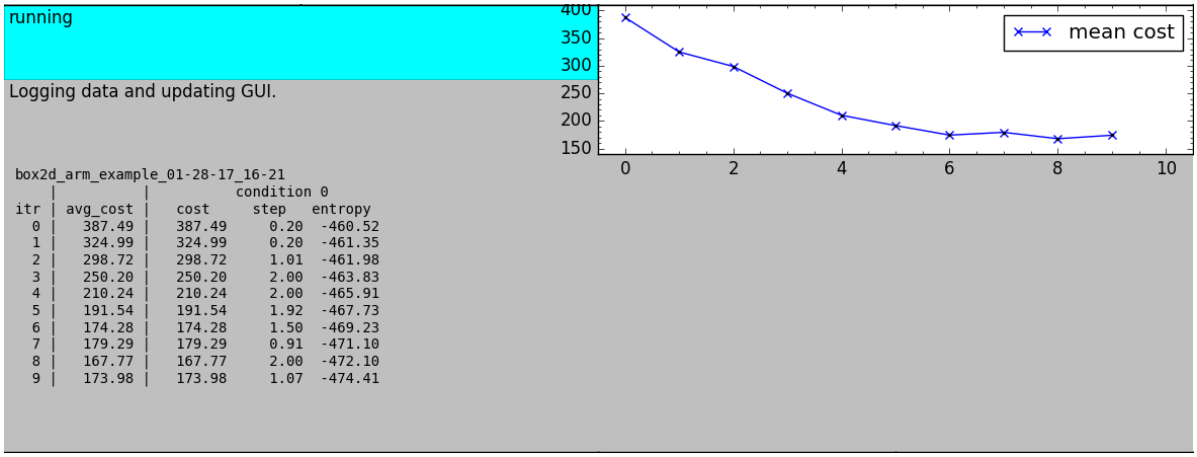
1.2. Above image is the result: Row from left to right: iteration, avg_cost, cost, step, entropy.

2. MPC unconstrained

2.1. Setup: 10 iterations, 1 condition (initial state), 5 samples, T: 100



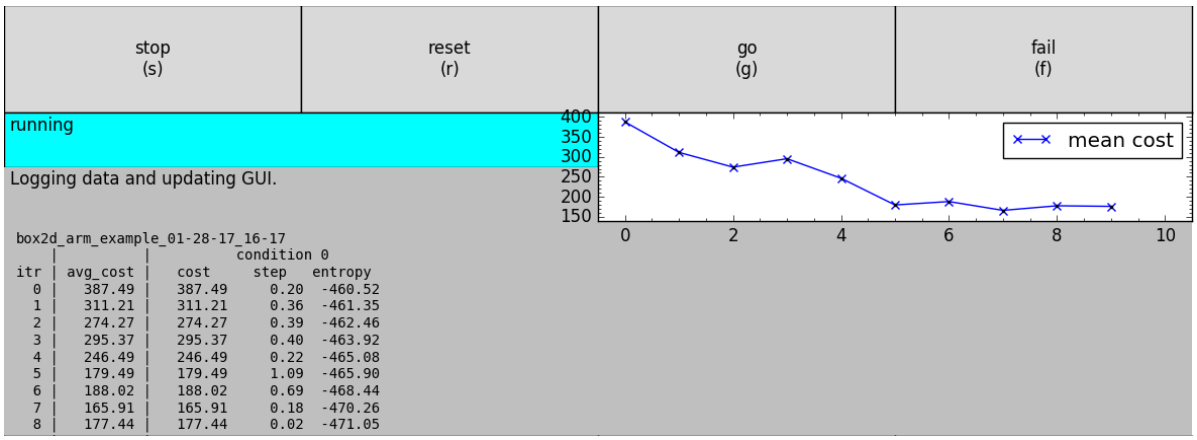
2.2. Short horizon: 10



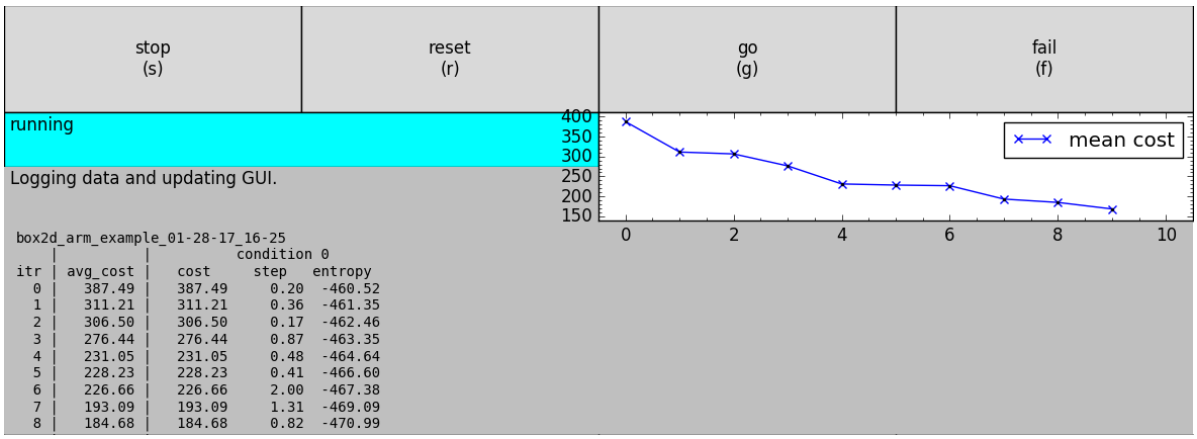
2.3. Short horizon: 5

3. MPC constrained (solved by DGD)

3.1. Setup: same as MPC unconstrained



3.2. Short horizon: 10



3.3. Short horizon: 5

IV. Question

1. Why it work when using surrogate cost $l = -log\ q(x_{t'}|x_t) - log\ p(u_{t'}|x_{t'})$ without DGD?

   Draft Answer: Because if I added $-log\ p(u_{t'}|x_{t'})$. The problem become:
   $\min_{q(u_{t'}|x_{t'})} E_{p(x_t)}[-log\ p(x_{t'}|x_t)] + D_{KL}(q(u_{t'}|x_{t'})\|p(u_{t'}|x_{t'}))$ (KL divergence between MPC policy and offline LQG policy), it is unconstrained problem, and two term is use log → has the same scale, so D_{KL} has weight = 1.0 is fine (Just a prediction – not sure).

2. If I mimic exactly the same as offline LQG, then the problem become:
   $\min_{q(u_{t'}|x_{t'})} E_{p(x_t)}[-log\ p(x_{t'}|x_t)] + \eta D_{KL}(q(u_{t'}|x_{t'})\|p(u_{t'}|x_{t'})) - \eta\epsilon$
   It equivalent to a constraint problem $D_{KL}(q(u_{t'}|x_{t'})\|p(u_{t'}|x_{t'})) \leq \epsilon$, it work but the cost is not lower (better) than above procedure?
   Note: To compute predicted cost and actual cost (Check Dual step_size of KL divergence constraint) I used the same procedure and cost is raw cost from offline LQG, NOT SURROGATE cost (I tried to use surrogate cost and estimate expected previous cost l_{k-1}^{k-1} is too far from computed previous cost).

   Draft Answer: Is this too hard constraint or my way to compute predicted and actual cost to adjust step size is wrong in this case ??

3. Why in the source code, it said Laplace approximation? I though Laplace approximation use to estimate distribution $p(z)$ with a Gaussian of distribution, $p(z) \approx q(z) = \mathcal{N}(z_0,\ A^{-1})$, where $A = \dfrac{\partial^2}{\partial z^2} ln\ p(z)$ is Hessian around z_0 of $ln\ p(z)$.

V. Future work

1. In the box2d_pointmass world, test more obstacle weight of cost.
2. Demystify BADMM for training neural network policy.
3. Read more on Information Theory.