Proof of Parallel MPPI

February 2, 2024

1 MPPI Review

The fundamental implementation of MPPI is summarized as follows (excerpted from NJSDF), At each control step:

a. Generate exploratory trajectories: Sample a batch of control sequences $\{u_{i,h}\}_{i=1\ldots N}^{h=1..H}$ of size $N \times H$ from the current distribution π_{t-1}. Roll out N trajectories $\{x_{i,h}\}_{i=1..N}^{h=1..H}$ of length H using dynamic model.

$$\pi_{t-1} = \prod_{h=1}^{H} \pi_{t-1,h}, \text{ where } \pi_{t-1,h} = \mathcal{N}(\mu_{t-1,h}, \Sigma_{t-1,h}) \tag{1}$$

b. Compute Cost Function: Design corresponding cost functions based on the policy, calculate the cost for each state$\{x_{i,h}\}_{i=1..N}^{h=1..H}$, and obtain the total cost for N trajectories:

$$\text{Cost}(i) = \sum_{h=0}^{H} \text{Cost}(i,h), \quad i = 1, 2, \ldots, N \tag{2}$$

c. Update Mean and Covariance: Utilize the softmax function to compute the weight $w(i)$ of each trajectory. Calculate the next-step policy $\mathcal{N}(\mu_{t,h}, \Sigma_{t,h})$ based on the weights and control sequences.

$$w(i) = \text{softmax}\left(-\frac{1}{\lambda} \cdot \text{Cost}(i)\right) \tag{3}$$

$$\hat{\mu}_{t,h} = (1 - \alpha_\mu)\hat{\mu}_{t-1,h} + \alpha_\mu \sum_{i=1}^{N} w_i u_{i,h} \tag{4}$$

$$\hat{\Sigma}_{t,h} = (1 - \alpha_\sigma)\hat{\Sigma}_{t-1,h} + \alpha_\sigma \sum_{i=1}^{N} w_i (u_{i,h} - \hat{\mu}_{t,h})(u_{i,h} - \hat{\mu}_{t,h})^T \tag{5}$$

2 Proof of Parallel MPPI

The formula for MME-DDP is provided as follows:

$$\pi(u|x) = \sum\nolimits_{n=1}^{N} w^{(n)}(x)\pi^{(n)}(u|x) \tag{1.1}$$

where

$$\pi^{(n)}(\delta u^{(n)}|\delta x^{(n)}) = \mathcal{N}(\delta u^{(n)}; \delta u^{(n)*}, \alpha(Q_{uu}^{(n)})^{-1}) \tag{1.2}$$

$$\begin{aligned} w^{(n)}(t, x_t) &:= z_t(t, x)^{-1} z_t^{(n)}(t, x) \\ &= \frac{\exp\left(-\frac{1}{\alpha}\left[V^{(n)}(t, x_t)\right]\right)}{\sum_{n'=1}^{N} \exp\left(-\frac{1}{\alpha}\left[V^{(n)}(t, x_t)\right]\right)} \end{aligned} \tag{1.3}$$

The evaluation of paths planned by each MPPI with different policies is conducted under a unified evaluation standard provided by the Judge Policy.

$$\mu_{t,h} = (1 - \alpha_\mu)\mu_{t-1,h} + \alpha_\mu \sum\nolimits_{i=1}^{N_{\text{policy}}} w_i \pi^i(u|x) \tag{1}$$

$$\Sigma_{t,h} = (1 - \alpha_\sigma)\mu_{t-1,h} + \alpha_\sigma \sum\nolimits_{i=1}^{N_{\text{policy}}} w_i \Sigma_{t,h}^i \tag{2}$$

$$w_i = \frac{\exp(-\frac{1}{\alpha}(V_{\text{policy}(i)} - V_{\text{policy}(i)}.\min()))}{\sum_{i=1}^{N_{\text{policy}}} \exp(-\frac{1}{\alpha}(V_{\text{policy}(i)} - V_{\text{policy}(i)}.\min()))} \tag{3}$$

The formula for MPQ is given as:

$$V_{\text{policy}(i)} = -\lambda \log E\left[\exp\left(-\frac{1}{\lambda}\Big(\sum_{l=1}^{H-1} c_{\pi^*}(s_{t+l}, a_{t+l}) + Q^{\pi^*}(s_{t+H}, a_{t+H})\Big)\right)\right] \tag{4}$$

$\sum_{l=1}^{H-1} c_{\pi^*}(s_{t+l}, a_{t+l})$ denotes the cost function of trajectories sampled from policy π_i but evaluated on the judge policy π^*. Ignore $Q^{\pi^*}(s_{t+H}, a_{t+H})$ and let $\text{Cost}^{\pi^*}(i) := \sum_{l=1}^{H} c(s_{t+l}, a_{t+l})$ for simplicity.

$$\begin{aligned} V_{\text{policy}(i)} &= -\lambda \log E\left[\exp\left(-\frac{1}{\lambda}\left(\sum\nolimits_{l=1}^{H} c_{\pi^*}(s_{t+l}, a_{t+l})\right)\right)\right] \\ &= -\lambda \log\left(\frac{1}{N}\sum\nolimits_{l=1}^{N} \exp\left(-\frac{1}{\lambda}\text{Cost}^{\pi^*}(i)\right)\right) \end{aligned} \tag{5}$$

Using policy-dependent weight: $\frac{\exp\left(-\frac{1}{\beta}\text{Cost}^{\pi_i}(i)\right)}{\sum_{l=1}^{N}\exp\left(-\frac{1}{\beta}\text{Cost}^{\pi_i}(i)\right)}$ rather than uniform weights $\frac{1}{N}$ allows for a better quantification of differences between policies π^i.

$$
\begin{aligned}
V_{\text{policy}(i)} &= -\lambda \log \left(\sum_{l=1}^{N} \frac{\exp\left(-\frac{1}{\beta}\text{cost}^{\pi_i}(i)\right)}{\sum_{l=1}^{N}\exp\left(-\frac{1}{\beta}\text{cost}^{\pi_i}(i)\right)} \exp\left(-\frac{1}{\lambda}\text{cost}^{\pi*}(i)\right) \right) \\
&= -\lambda \log \left(\left(\frac{1}{\sum_{l=1}^{N}\exp\left(-\frac{1}{\beta}\text{cost}^{\pi_i}(i)\right)} \right) \sum_{l=1}^{N} \exp\left(\left(-\frac{1}{\beta}\text{cost}^{\pi_i}(i)\right) + \left(-\frac{1}{\lambda}\text{cost}^{\pi*}(i)\right) \right) \right) \\
&= -\lambda \log \sum_{l=1}^{N} \exp\left(\left(-\frac{1}{\beta}\text{cost}^{\pi_i}(i)\right) + \left(-\frac{1}{\lambda}\text{cost}^{\pi*}(i)\right) \right) + \lambda \log \sum_{l=1}^{N} \exp\left(-\frac{1}{\beta}\text{cost}^{\pi_i}(i)\right)
\end{aligned} \tag{6}
$$

to avoid potential numerical overflow, let

$$
A_l = -1/\beta \text{cost}\left(t^{\pi_i}(i)\right) - 1/\lambda \text{cost}\left(t^{\pi*}(i)\right)
$$

$$
B_l = -1/\beta \text{cost}\left(t^{\pi_i}(i)\right)
$$

$$
V_{\text{policy}(i)} = -\lambda \left(\log \sum\nolimits_{l=1}^{N} \exp\left(A_l - A_{\max}\right) + \log \sum\nolimits_{l=1}^{N} \exp\left(B_l - B_{\max}\right) - A_{\max} + B_{\max} \right) \tag{7}
$$

2.1 the influence of α,λ,β

α, λ, and β are all used in the policy evaluation step of the Parallel MPPI algorithm, but they serve different purposes. The β parameter is used to calculate the weights of the $TopN$ trajectories given to the Judge Policy by different policies. It can more fully extract the intrinsic differences between different policies. On the other hand, the α and λ parameters only affect the formula for calculating the expected value and do not affect the calculation of the weight function, so they cannot control the differences in weights between different policies.

Using the softmax function with the parameter β replaces the original use of uniform weights $1/N$. Here, the role of β is:

1. By changing the value of β, you can control the "softness" of the softmax function.

2. The larger the β value, the more uniform the softmax distribution becomes. Differences between function values are less pronounced, and it tends to favor $1/N$.

3. The smaller the β value, the narrower the softmax distribution becomes. Differences between function values are more pronounced, but policy evaluation becomes more extreme.

When β is adjusted to an appropriate value, it can better capture the differences between policies. Therefore, the β parameter can finely control the weight distribution formed by the softmax function, providing a tunable hyperparameter that makes policy-oriented weights more closely resemble actual policy differences.

2.2 why Judge Policy

The Judge Policy provides an independent and objective evaluation standard, enabling the quantification (scoring) and comparison (assessment) of trajectories planned by various policies under the same criteria. This approach avoids direct comparisons using internal standards of each policy. Regardless of the differences in internal standards among policies, they are assessed within a consistent, neutral framework. The core purpose of designing the Judge Policy is to ensure fairness in the evaluation and comparison of different policies, without being influenced by individual policy-specific standards.

2.3 the pseudo code of Parallel MPPI

Algorithm 1 Parallel MPPI

Given: Parallel MPPI Parameters $topN$, β, λ, α
while task not complete **do**
  Sample M trajs. $\theta_{M,t}$ of len. H from $\mathcal{N}(\mu, \widehat{\Sigma}_{t-1})$, $\mu \in \{\widehat{\mu}_{t-1}, \widehat{\mu}_{\pi_i,t-1}\}$
  // Evaluate different policies (e.g., Greedy and Sensitive)
  for each π_i **do**
    for $k = 1$ to M in parallel **do**
      $Cost^{\pi_i}(\theta_k) := \sum_{l=1}^{H} c^{\pi_i}(\theta_{k,l})$
    end for
    $\widehat{\mu}_{\pi_i,t}, \widehat{\Sigma}_{\pi_i,t}$:= MPPI_Base_Function Equation 1
    // π_i selects topN trajs and assigns weight for each traj.
    topN_index $\leftarrow$ sort$(Cost^{\pi_i}(\theta_M), topN)$
    $\theta_{topN} := \theta_M$(topN_index)
    $w_{\theta_{topN}} := \frac{\exp\left(-\frac{1}{\beta}\text{Cost}^{\pi_i}(\theta_{topN})\right)}{\sum_{l=1}^{topN}\exp\left(-\frac{1}{\beta}\text{Cost}^{\pi_i}(\theta_{topN})\right)}$
    // MPPI Value Function
    $V_{\pi_i} := E[Cost^{\text{Judge\_Policy}}(\theta_{topN}, w_{\theta_{topN}}, \lambda)]$ // Equation 6
  end for
  // Update mixed Gaussian weight using $V_{\text{policy}(i)}$
  for each π_i **do**
    $w^{\pi_i} := \text{Softmax}(-\frac{1}{\alpha}V_{\pi_i})$
  end for
  $\widehat{\mu}_t := (1-\alpha_\mu)\widehat{\Sigma}_{t-1} + \alpha_\mu \sum_{i=1}^{N_{\text{policy}}} w^{\pi_i}\widehat{\mu}_{\pi_i,t}$
  $\widehat{\Sigma}_t := (1-\alpha_\sigma)\widehat{\Sigma}_{t-1} + \alpha_\sigma \sum_{i=1}^{N_{\text{policy}}} w^{\pi_i}\widehat{\Sigma}_{\pi_i,t}$
end while

In this section, we introduce the Parallel MPPI algorithm.

1. Line 3 samples a batch of control sequences $\{u_{i,h}\}_{i=1\ldots M}^{h=1..H}$ of size $M \times H$ from $\mathcal{N}(\mu, \widehat{\Sigma}_{t-1})$. Roll out the dynamic model to obtain M trajectories $\theta_{M,t}$ of length H.

Remark. The mean μ includes $\widehat{\mu}_{\pi_i,t-1}$ generated by the Greedy and Sensitive policies, as well as the $\widehat{\mu}_{t-1}$, which is the combined result of control sequences from different policies. The M trajectories $\theta_{M,t}$ are divided into N trajectories $\theta_{N,t}, \theta_{N\sim 2N,t}$ for each $\widehat{\mu}_{\pi_i,t-1}$ and $M-2N$ trajectories $\theta_{M-2N,t}$ for $\widehat{\mu}_{t-1}$.

2. Lines 5 to 15 evaluate different policies (e.g., Greedy and Sensitive)

More specifically, Lines 6 to line 9 run the basic MPPI algorithm, computing the cost function $Cost^{\pi_i}(\theta_M)$ designed by the corresponding policy and utilizing the softmax to obtain $\widehat{\mu}_{\pi_i,t}, \widehat{\Sigma}_{\pi_i,t}$.

Lines 10 to 15 determine the evaluation results for each policy. The policy selects the top N trajectories θ_{topN} with the lowest cost values and assigns confidence weights $w_{\theta_{topN}}$ to each trajectory. These trajectories are then collectively submitted to the Judge Policy, which calculates scores based on the derived evaluation formula (Equation 6).

3. Lines 18 to 20 compute Policy Weight. Based on the value function V_{π_i} corresponding to each policy, the softmax is used to assign weights for each policy.

4. Lines 21 to 23 update the mixed mean $\widehat{\mu}_t$ and covariance $\widehat{\Sigma}_t$ based on the policy weights.

This comprehensive process of sampling, evaluating, and updating iterates until the completion of the task, providing an adaptive and efficient approach to trajectory planning in the context of multiple policies.

3 brief summary