A Generalized Path Integral Control Approach to Reinforcement Learning

Yunian Pan

ECE department

March. 29th 2019

From Stochastic Optimal control to Reinforcement Learning

- Theoretical development
- RL with Parameterized Policies
- Evaluations
- Conclusion

Policy Improvement with Path Integrals (PI^2) offers currently one of the most efficient, numerically robust, and easy to implement algorithms for RL based on trajectory roll-outs.

Problem Construction

- The cost: $R(\tau_i) = \Phi_{t_N} + \int_{t_i}^{t_N} r_t dt$

Problem Construction

- The cost: $R(\tau_i) = \Phi_{t_N} + \int_{t_i}^{t_N} r_t dt$
- The optimization problem: $V(x_{t_i}) = V_{t_i} = \min_{u_{t_i:t_N}} E_{\tau_i}[R(\tau_i)]$

Problem Construction

- The cost: $R(\tau_i) = \Phi_{t_N} + \int_{t_i}^{t_N} r_t dt$
- The optimization problem: $V(x_{t_i}) = V_{t_i} = \min_{u_{t_i:t_N}} E_{\tau_i}[R(\tau_i)]$
- A stochastic dynamic constraint:
  $\dot{x}_t = f(x_t, t) + G(x_t)(u_t + \epsilon_t) = f_t + G_t(u_t + \epsilon_t)$
  Where the immediate cost r_t depends respectively on states and commands: $r_t = r(x_t, u_t, t) = q_t + \frac{1}{2} u_t^\top R u_t$

Problem Construction

- The cost: $R(\tau_i) = \Phi_{t_N} + \int_{t_i}^{t_N} r_t dt$
- The optimization problem: $V(x_{t_i}) = V_{t_i} = \min_{u_{t_i:t_N}} E_{\tau_i}[R(\tau_i)]$
- A stochastic dynamic constraint:
  $\dot{x}_t = f(x_t, t) + G(x_t)(u_t + \epsilon_t) = f_t + G_t(u_t + \epsilon_t)$
  Where the immediate cost r_t depends respectively on states and commands: $r_t = r(x_t, u_t, t) = q_t + \frac{1}{2} u_t^\top R u_t$

Resulting stochastic HJB equation:

$$-\partial_t V_t = \min_u (r_t + (\nabla_x V_t)^\top F_t + \frac{1}{2} trace((\nabla_{xx} V_t) G_t \Sigma_\epsilon G_t^\top))$$

How is the HJB derived?

- By Bellman's principle of optimality, within a small dt:
  $V_t = \min E[V_{t+dt} + \int_t^{t+dt} r_t dt]$

How is the HJB derived?

- By Bellman's principle of optimality, within a small dt:
  $V_t = \min E[V_{t+dt} + \int_t^{t+dt} r_t dt]$
- Using second order Taylor expansion:
  $-\partial_t V_t = \min E[r_t + (\nabla_x V_t)\dot{x} + \dot{x}^\top (\nabla_{xx} V_t)\dot{x} dt]$

How is the HJB derived?

- By Bellman's principle of optimality, within a small dt:
  $V_t = \min E[V_{t+dt} + \int_t^{t+dt} r_t dt]$
- Using second order Taylor expansion:
  $-\partial_t V_t = \min E[r_t + (\nabla_x V_t)\dot{x} + \dot{x}^\top (\nabla_{xx} V_t)\dot{x}dt]$
- With partition of the stochastic dynamics and some nice property of Gaussian white noise:
  $f(x_t, t) + G(x_t)u_t = F_t$, $E(\epsilon_t^\top \epsilon_{t+dt})dt \approx \Sigma_{\epsilon_t}$, $E(\epsilon_t) = 0$, cancel dt.

How is the HJB derived?

- By Bellman's principle of optimality, within a small dt:
  $V_t = \min E[V_{t+dt} + \int_t^{t+dt} r_t dt]$
- Using second order Taylor expansion:
  $-\partial_t V_t = \min E[r_t + (\nabla_x V_t)\dot{x} + \dot{x}^\top(\nabla_{xx} V_t)\dot{x}dt]$
- With partition of the stochastic dynamics and some nice property of Gaussian white noise:
  $f(x_t, t) + G(x_t)u_t = F_t$, $E(\epsilon_t^\top \epsilon_{t+dt})dt \approx \Sigma_{\epsilon_t}$, $E(\epsilon_t) = 0$, cancel dt.
- Introducing trace operator to apply cyclic property s.t. it can be written as quadratic form

Not done yet

1. Inserting the immediate cost in it to take deravative w.r.t. u_t

Not done yet

1. Inserting the immediate cost in it to take deravative w.r.t. u_t
2. set it 0 to get the optimal control $u(x_t) = u_t = -R^{-1}G_t^\top(\nabla_{x_t}V_t)$

Not done yet

1. Inserting the immediate cost in it to take deravative w.r.t. u_t
2. set it 0 to get the optimal control $u(x_t) = u_t = -R^{-1}G_t^\top(\nabla_{x_t} V_t)$
3. insert the optimal control back into HJB equation to remove $\min$

Not done yet

1. Inserting the immediate cost in it to take deravative w.r.t. u_t
2. set it 0 to get the optimal control $u(x_t) = u_t = -R^{-1}G_t^\top(\nabla_{x_t} V_t)$
3. insert the optimal control back into HJB equation to remove min

Result

Now we get a second order nonlinear PDE:

$$-\partial_t V_t = q_t + (\nabla_x V_t)^\top f_t - \frac{1}{2}(\nabla_x V_t)^\top G_t R^{-1} G_t^\top(\nabla_{x_t} V_t) + \frac{1}{2} trace((\nabla_{xx} V_t) G_t \Sigma_\epsilon G_t^\top)$$

from Nonlinear to Linear PDE

Using a logarithmic transformation of the value function

$$V_t = -\lambda \log \Psi_t$$

from Nonlinear to Linear PDE

Using a logarithmic transformation of the value function

$$V_t = -\lambda \log \Psi_t$$

Making a strong assumption that $\lambda R^{-1} = \Sigma_\epsilon$ to get a linear PDE

$$-\partial_t \Psi_t = -\frac{1}{\lambda} q_t \Psi_t + f_t^\top (\nabla_x \Psi_t) + \frac{1}{2} trace((\nabla_{xx} \Psi_t) G_t \Sigma G_t^\top)$$

from Nonlinear to Linear PDE

Using a logarithmic transformation of the value function

$$V_t = -\lambda \log \Psi_t$$

Making a strong assumption that $\lambda R^{-1} = \Sigma_\epsilon$ to get a linear PDE

$$-\partial_t \Psi_t = -\frac{1}{\lambda} q_t \Psi_t + f_t^\top (\nabla_x \Psi_t) + \frac{1}{2} trace((\nabla_{xx} \Psi_t) G_t \Sigma G_t^\top)$$

Applying Feynman-Kac lemma to solve the linear PDE

$$\Psi_{t_i} = \lim_{dt \to 0} \int p(\tau_i | x_{t_i}) \exp[-\frac{1}{\lambda}(\phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} dt)] d\tau_i$$

Path integral formulation

- states partitioned as non-actuated and actuated parts:
$$\begin{pmatrix} x^{(m)}_{t_{i+1}} \\ x^{(c)}_{t_{i+1}} \end{pmatrix} = \begin{pmatrix} x^{(m)}_{t_i} \\ x^{(c)}_{t_i} \end{pmatrix} + \begin{pmatrix} f^{(m)}_{t_i} \\ f^{(c)}_{t_i} \end{pmatrix} dt + \begin{pmatrix} 0_{k\times p} \\ G^{(c)}_{t_i} \end{pmatrix} (u_{t_i} dt + \sqrt{dt}\epsilon_{t_i})$$

Path integral formulation

- states partitioned as non-actuated and actuated parts:
$$\begin{pmatrix} x^{(m)}_{t_{i+1}} \\ x^{(c)}_{t_{i+1}} \end{pmatrix} = \begin{pmatrix} x^{(m)}_{t_i} \\ x^{(c)}_{t_i} \end{pmatrix} + \begin{pmatrix} f^{(m)}_{t_i} \\ f^{(c)}_{t_i} \end{pmatrix} dt + \begin{pmatrix} 0_{k\times p} \\ G^{(c)}_{t_i} \end{pmatrix} (u_{t_i} dt + \sqrt{dt}\epsilon_{t_i})$$
- transition probability factorization (Markov property)

$$p(\tau_i | x_i) = \prod_{j=i}^{N-1} p(x_{t_{j+1}} | x_{t_j}) \propto \prod_{j=i}^{N-1} p(x^{(c)}_{t_{j+1}} | x_{t_j})$$

Path integral formulation

The states are determined by the dynamics With Gaussian noise:

$$p(x^{(c)}_{t_j+1}|x_{t_j}) = \frac{1}{((2\pi)^l \cdot |\Sigma_{t_j}|)^{\frac{1}{2}}} \exp(-||x^{(c)}_{t_j+1} - x^{(c)}_{t_j} - f^{(c)}_{t_j} dt||^2_{\Sigma_{t_j}})$$

Incorporate with previous assumption $\Sigma_{t_j} = G^{(c)}_{t_j} \Sigma_\epsilon {G^{(c)}_{t_j}}^\top dt = \lambda G^{(c)}_{t_j} R^{-1} {G^{(c)}_{t_j}}^\top dt = \lambda H_{t_j} dt$, and insert back:

Path integral formulation

The states are determined by the dynamics With Gaussian noise:

$$p(x^{(c)}_{t_j+1}|x_{t_j}) = \frac{1}{((2\pi)^l \cdot |\Sigma_{t_j}|)^{\frac{1}{2}}} \exp(-||x^{(c)}_{t_j+1} - x^{(c)}_{t_j} - f^{(c)}_{t_j} dt||^2_{\Sigma_{t_j}})$$

Incorporate with previous assumption $\Sigma_{t_j} = G^{(c)}_{t_j} \Sigma_\epsilon G^{(c)}_{t_j}{}^\top dt = \lambda G^{(c)}_{t_j} R^{-1} G^{(c)}_{t_j}{}^\top dt = \lambda H_{t_j} dt$, and insert back:

$$\Psi_{t_i} = \lim_{dt \to 0} \int \exp(-\frac{1}{\lambda} S(\tau_i) - \log D(\tau_i)) d\tau_i^{(c)}$$

$S(\tau_i) = \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} dt + \frac{1}{2} \sum_{j=i}^{N-1} ||\frac{x^{(c)}_{t_j+1} - x^{(c)}_{t_j}}{dt} - f^{(c)}_{t_j}||^2_{H_{t_j}} dt$

$D(\tau_i) = \prod_{j=i}^{N-1} ((2\pi)^l \cdot |\Sigma_{t_j}|)^{\frac{1}{2}})$

Deriving Optimal Control

- substitute the Jacobian $\nabla_{x_{t_i}} V_{t_i}$: $u_{t_i} = \lambda R^{-1} G_{t_i} \frac{\nabla_{x_{t_i}} \Psi_{t_i}}{\Psi_{t_i}}$

Deriving Optimal Control

- substitute the Jacobian $\nabla_{x_{t_i}} V_{t_i}$: $u_{t_i} = \lambda R^{-1} G_{t_i} \frac{\nabla_{x_{t_i}} \Psi_{t_i}}{\Psi_{t_i}}$
- Probability matching: $u_{t_i} = \int P(\tau_i) u_L(\tau_i) d\tau_i^{(c)}$

  Where local probability: $P(\tau_i) = \frac{e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)}}{\int e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)} d\tau_i}$

  local control: $u_L(\tau_i) = -R^{-1} G_{t_i}^{(c)^\top} \lim_{dt\to 0}(\nabla_{x_{t_i}^{(c)}} \tilde{S}(\tau_i))$

Deriving Optimal Control

- substitute the Jacobian $\nabla_{x_{t_i}} V_{t_i}$: $u_{t_i} = \lambda R^{-1} G_{t_i} \frac{\nabla_{x_{t_i}} \Psi_{t_i}}{\Psi_{t_i}}$
- Probability matching: $u_{t_i} = \int P(\tau_i) u_L(\tau_i) d\tau_i^{(c)}$

  Where local probability: $P(\tau_i) = \frac{e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)}}{\int e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)} d\tau_i}$

  local control: $u_L(\tau_i) = -R^{-1} G_{t_i}^{(c)\top} \lim_{dt \to 0}(\nabla_{x_{t_i}^{(c)}} \tilde{S}(\tau_i))$
- compute the Jacobian of local cost: $\nabla_{x_{t_i}^{(c)}} \tilde{S}(\tau_i)$

Deriving Optimal Control

- substitute the Jacobian $\nabla_{x_{t_i}} V_{t_i}$: $u_{t_i} = \lambda R^{-1} G_{t_i} \frac{\nabla_{x_{t_i}} \Psi_{t_i}}{\Psi_{t_i}}$
- Probability matching: $u_{t_i} = \int P(\tau_i) u_L(\tau_i) d\tau_i^{(c)}$

  Where local probability: $P(\tau_i) = \frac{e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)}}{\int e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)} d\tau_i}$

  local control: $u_L(\tau_i) = -R^{-1} {G_{t_i}^{(c)}}^\top \lim_{dt\to 0}(\nabla_{x_{t_i}^{(c)}} \tilde{S}(\tau_i))$
- compute the Jacobian of local cost: $\nabla_{x_{t_i}^{(c)}} \tilde{S}(\tau_i)$
- Lead to our final local optimal control:

$$u_L(\tau_i) = R^{-1} {G_{t_i}^{(c)}}^\top (G_{t_i}^{(c)} R^{-1} {G_{t_i}^{(c)}}^\top)^{-1} (G_{t_i}^{(c)} \epsilon_{t_i} - b_{t_i})$$

- Given:
  - the system dynamics: $\dot{x}_t = f(x_t, t) + G(x_t)(u_t + \epsilon_t) = f_t + G_t(u_t + \epsilon_t)$
  - The immediate cost: $r_t = r(x_t, u_t, t) = q_t + \frac{1}{2} u_t^\top R u_t$
  - A terminal cost term Φ_{t_N}
  - The variance Σ_ϵ of the mean-zero noise ϵ_t
  - Trajectory starting at t_i and ending at t_N : $\tau_i = (x_{t_i}, \dots, x_{t_N})$
  - A partitioning of the system dynamics into (c) controlled and (m) uncontrolled equations, where n = c + m is the dimensionality of the state x_t
- Optimal Controls:
  - optimal controls at every time step t_i: $u_{t_i} = \int P(\tau_i) u_L(\tau_i) d\tau_i^{(c)}$
  - Probability of a trajectory: $P(\tau_i) = \frac{e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)}}{\int e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)} d\tau_i}$
  - Generalized trajectory cost: $\tilde{S}(\tau_i) = S(\tau_i) + \frac{\lambda}{2} \sum_{j=i}^{N-1} \log |H_{t_j}|$
  - Local controls: $u_L(\tau_i) = R^{-1} G_{t_i}^{(c)\top} (G_{t_i}^{(c)} R^{-1} G_{t_i}^{(c)\top})^{-1} (G_{t_i}^{(c)} \epsilon_{t_i} - b_{t_i})$

The probability of the trajectory:

$$p(\tau_i) = p(x_{t_i}) \prod_{j=i}^{N-1} p(x_{t_{j+1}} | x_{t_j}, \alpha_{t_j}) p(\alpha_{t_j} | x_{t_j})$$

The probability of the trajectory:

$$p(\tau_i) = p(x_{t_i}) \prod_{j=i}^{N-1} p(x_{t_{j+1}} | x_{t_j}, \alpha_{t_j}) p(\alpha_{t_j} | x_{t_j})$$

Usually for continuous state-action domains, the stochastic policies are parameterized as:

$$\alpha_{t_i} = g_{t_i}^{\top} (\theta + \epsilon_{t_i})$$

The probability of the trajectory:

$$p(\tau_i) = p(x_{t_i}) \prod_{j=i}^{N-1} p(x_{t_{j+1}} | x_{t_j}, \alpha_{t_j}) p(\alpha_{t_j} | x_{t_j})$$

Usually for continuous state-action domains, the stochastic policies are parameterized as:

$$\alpha_{t_i} = g_{t_i}^{\top} (\theta + \epsilon_{t_i})$$

g_{t_i}: a vector of basis functions.
$p(\alpha_{t_i} | x_{t_i}) = N(\theta^{\top} g_{t_i}, \Sigma_{t_i})$ with $\Sigma_{t_i} = g_{t_i}^{\top} \Sigma_{\epsilon} g_{t_i}$

DMPs

$$
\begin{aligned}
\frac{1}{\tau}\dot{z}_t &= f_t + g_t^\top(\theta + \epsilon_t) \\
\frac{1}{\tau}\dot{y}_t &= z_t \\
\frac{1}{\tau}\dot{x}_t &= -\alpha x_t \\
f_t &= \alpha_z(\beta_z(g - y_t) - z_t).
\end{aligned}
$$

DMPs

$$\frac{1}{\tau}\dot{z}_t = f_t + g_t^\top(\theta + \epsilon_t)$$
$$\frac{1}{\tau}\dot{y}_t = z_t$$
$$\frac{1}{\tau}\dot{x}_t = -\alpha x_t$$
$$f_t = \alpha_z(\beta_z(g - y_t) - z_t).$$

g_{t_i}: piecewise linear function approximator with Gaussian weighting kernels c_j's

$$|g_t|_j = \frac{w_j x_t}{\sum_{k=1}^{p} w_k}(g - y_{t_0})$$
$$w_j = \exp(0.5 h_j (x_t - c_j)^2)$$

From DMPs to PI^2

Building blocks:

$$\begin{pmatrix} \dot{x}_t \\ \dot{y}_t \\ \dot{z}_t \end{pmatrix} = \begin{pmatrix} -\alpha x_t \\ z_t \\ \alpha_z(\beta_z(g - y_t) - z_t) + g_t^\top(\theta + \epsilon_t) \end{pmatrix} + \begin{pmatrix} 0_{1\times p} \\ 0_{1\times p} \\ g_{t_i}^{(c)\top} \end{pmatrix} (\theta_t + \epsilon_t)$$

From DMPs to PI^2

Building blocks:

$$\begin{pmatrix} \dot{x}_t \\ \dot{y}_t \\ \dot{z}_t \end{pmatrix} = \begin{pmatrix} -\alpha x_t \\ z_t \\ \alpha_z(\beta_z(g - y_t) - z_t) + g_t^\top(\theta + \epsilon_t) \end{pmatrix} + \begin{pmatrix} 0_{1\times p} \\ 0_{1\times p} \\ {g_{t_i}^{(c)}}^\top \end{pmatrix} (\theta_t + \epsilon_t)$$

Computing path cost:

$$S(\tilde{\tau}_i) = \phi_{t_N} + \sum_{j=i}^{N-1} q_{t_j} + \frac{1}{2}\sum_{j=i}^{N-1} \frac{1}{2}(\theta_{t_j} + \epsilon_{t_j})^\top {M_{t_j}}^\top R^{-1} M_{t_j}(\theta_{t_j} + \epsilon_{t_j})$$

With projection matrix $M_{t_j} = \frac{R^{-1} g_{t_j} g_{t_j}^\top}{g_{t_j}^\top R^{-1} g_{t_j}}$

From DMPs to PI^2

Update derivation:

$$
\begin{aligned}
\theta_{t_i}^{(new)} &= \int P(\tau_i) \frac{R^{-1} g_{t_j} g_{t_j}^\top}{g_{t_j}^\top R^{-1} g_{t_j}} (\theta + \epsilon_{t_i}) d\tau_i \\
&= \int P(\tau_i) \frac{R^{-1} g_{t_j} g_{t_j}^\top}{g_{t_j}^\top R^{-1} g_{t_j}} \epsilon_{t_i} d\tau_i + M_{t_i}\theta \\
&= \delta\theta + M_{t_i}\theta
\end{aligned}
$$

Averaging over time:

From DMPs to PI^2

Update derivation:

$$\theta_{t_i}^{(new)} = \int P(\tau_i) \frac{R^{-1} g_{t_j} g_{t_j}^\top}{g_{t_j}^\top R^{-1} g_{t_j}} (\theta + \epsilon_{t_i}) d\tau_i$$

$$= \int P(\tau_i) \frac{R^{-1} g_{t_j} g_{t_j}^\top}{g_{t_j}^\top R^{-1} g_{t_j}} \epsilon_{t_i} d\tau_i + M_{t_i}\theta$$

$$= \delta\theta + M_{t_i}\theta$$

Averaging over time:(Remove the projection)

$$\theta^{(new)} = \frac{1}{N}\sum_{i=0}^{N-1} \delta\theta_{t_i} + \frac{1}{N}\sum_{i=0}^{N-1} \theta == \frac{1}{N}\sum_{i=0}^{N-1} \delta\theta_{t_i} + \theta$$

Elimenating parameter λ

$$\exp(-\frac{1}{\lambda}S(\tau_i)) = \exp(-h\frac{S(\tau_i) - \min S(\tau_i)}{\max S(\tau_i) - \min S(\tau_i)})$$

- Given:
  - The immediate cost: $r_t = r(x_t, u_t, t) = q_t + \frac{1}{2}\theta_t^\top R\theta_t$
  - A terminal cost term Φ_{t_N}
  - A stochastic parameterized policy $\alpha_{t_i} = g_{t_i}^\top(\theta + \epsilon_{t_i})$
  - The basis function g_{t_i} from the system dynamics
  - The variance Σ_ϵ of the mean zero noise ϵ_t
  - The initial parameter vector θ

- Repeat until convergence of the tracjectory cost S
  - Create K roll-outs of the system from the same start state x_0 using stochstic parameters $\theta + \epsilon_t$ at every time step
  - For $k = 1, \dots, K$, compute:
    - $P(\tau_{i,k}) = \frac{e^{-\frac{1}{\lambda}S(\tau_{i,k})}}{\sum_{k=1}^{K}[e^{-\frac{1}{\lambda}S(\tau_{i,k})}]}$
    - $S(\tau_{i,k}) = \phi_{t_N,k} + \sum_{j=i}^{N-1} q_{t_j,k} + \frac{1}{2}\sum_{j=i}^{N-1} \frac{1}{2}(\theta + M_{t_j,k}\epsilon_{t_j,k})^\top R^{-1}(\theta + M_{t_j,k}\epsilon_{t_j,k})$
    - $M_{t_j,k} = \frac{R^{-1} g_{t_j,k} g_{t_j,k}^\top}{g_{t_j,k}^\top R^{-1} g_{t_j,k}}$
  - Probability of a trajectory: $P(\tau_i) = \frac{e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)} d\tau_i^{(c)}}{\int e^{-\frac{1}{\lambda}\tilde{S}(\tau_i)} d\tau_i}$
  - Generalized trajectory cost: $\tilde{S}(\tau_i) = S(\tau_i) + \frac{\lambda}{2}\sum_{j=i}^{N-1} \log |H_{t_j}|$
  - Local controls: $u_L(\tau_i) = R^{-1} G_{t_i}^{(c)\top} (G_{t_i}^{(c)} R^{-1} G_{t_i}^{(c)\top})^{-1} (G_{t_i}^{(c)} \epsilon_{t_i} - b_{t_i})$

- For $i = 1, \ldots, (N-1)$,compute:
  - $\delta\theta_{t_i} = \sum_{k=0}^{K}[P(\tau_{i,k})M_{t_{i,k}}\epsilon_{t_{i,k}}]$
- Compute: $[\delta\theta]_j = \frac{\sum_{i=0}^{N-1}(N-i)w_{j,t_i}[\delta\theta_{t_i}]_j}{\sum_{i=0}^{N-1}(N-i)w_{j,t_i}}$
- Update: $\theta^{(new)} = \theta^{(old)} + \delta\theta$
- Create one noiseless roll-out to check the trajectory cost $R = \Phi_{t_N} + \sum_{i=0}^{N-1} r_{t_i}$. In case the noise cannot be turned off, that is, a stochastic system, multiple roll-outs need be averaged.

4 Algorithms are implemented in comparison with PI^2:

1. REINFORCE: taking derivative w.r.t. policy.
2. GPOMDP: improvement over REINFORCE.
3. eNAC: using Fisher Information Matrix to project the REINFORCE gradient onto a more effective update direction.
4. PoWER: probabilistic policy improvement method, restrictive.

Performances are evaluated on following learning tasks:

- 1 DOF Reaching task: 1 dimensional DMP reaching.
- 1 DOF via-point task: 1 dimensional DMP approximating target hitting.
- multi-DOF via-point task: multi-dimensional DMP approximating target hitting.
- robot simulation: testing pi^2 with a 12-degree-freedom robot dog.

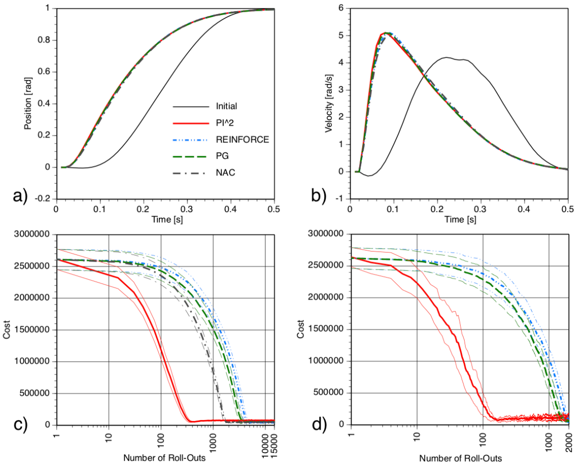


Figure 1: 1 DOF comparison

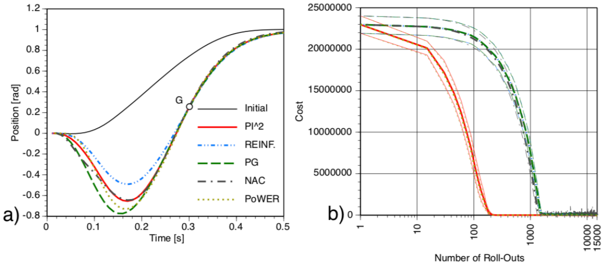


Figure 2: 1 DOF via-point comparison

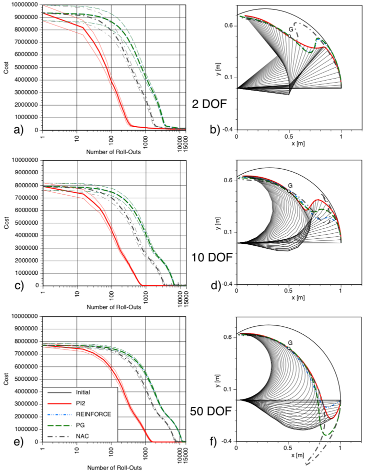
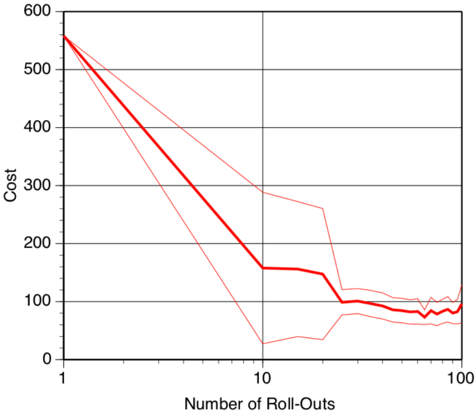


Figure 3: multi DOF via-point comparison



(b) Learning curve for Dog Jump with $\mathbf{PI}^2$ $\pm 1std$

Figure 4: learning curves

- The Simplification $\lambda R^{-1} = \Sigma_\epsilon$

- The Simplification $\lambda R^{-1} = \Sigma_\epsilon$
- Model-based, Hybrid, and Model-free Learning

- The Simplification $\lambda R^{-1} = \Sigma_\epsilon$
- Model-based, Hybrid, and Model-free Learning
- Rules of Cost Function Design

- The Simplification $\lambda R^{-1} = \Sigma_\epsilon$
- Model-based, Hybrid, and Model-free Learning
- Rules of Cost Function Design
- Hidden states

- The Simplification $\lambda R^{-1} = \Sigma_\epsilon$
- Model-based, Hybrid, and Model-free Learning
- Rules of Cost Function Design
- Hidden states
- In general, it’s good.

Thank You!