Deep Bayesian Quadrature Policy Optimization

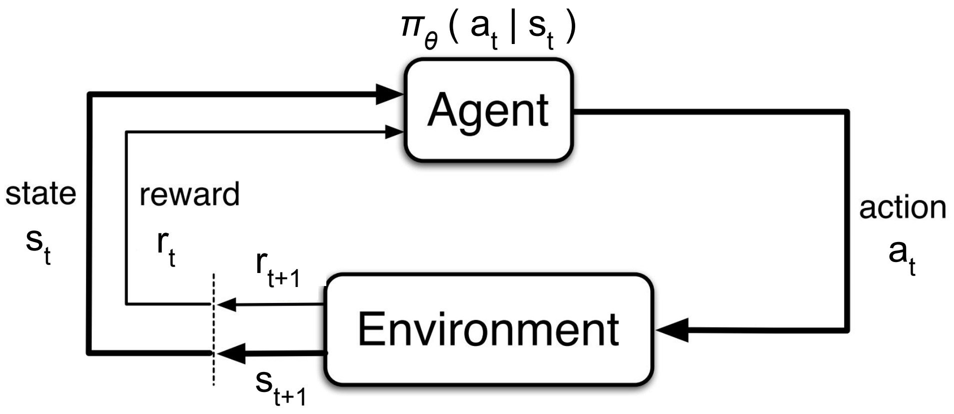
Akella Ravi Tej

joint work with
Kamyar Azizzadenesheli, Mohammad Ghavamzadeh, Anima Anandkumar, Yisong Yue

Overview

- Preliminaries
- Policy Gradient as Numerical Integration Problem
  - Monte-Carlo (MC) Estimation
  - Bayesian Quadrature (BQ)
- Deep Bayesian Quadrature Policy Gradient (DBQPG)
  - Scalable, sample-efficient policy gradient estimator.
- Uncertainty Aware Policy Gradient (UAPG)
  - Using the estimation uncertainty provided by DBQPG for reliable policy updates.
- Empirical Analysis

Preliminaries



The agent–environment interaction in a Markov decision process.

State-space	$s_t \in S$
Action-space	$a_t \in A$
Transition Kernel	$P : S \times A \to \Delta_S$
Reward Kernel	$r : S \times A \to \mathbb{R}$
Initial state distribution	$\rho_0 : S \to \Delta_S$
Stochastic Policy	$\pi_\theta : S \to \Delta_A$

Δ_S and Δ_A are distributions over S and A, respectively.

[Figure source: Sutton & Barto, 1998]

Useful Definitions

State-action pair: $z = (s, a)$

State-action transition dynamics: $P^{\pi_\theta}(z_t | z_{t-1}) = \pi_\theta(a_t | s_t) P(s_t | z_{t-1})$

Action-value function: $Q_{\pi_\theta}(z_t) = E\left[\sum_{\tau=0}^{\infty} \gamma^\tau r(z_{t+\tau}) \;\middle|\; z_{t+\tau+1} \sim P^{\pi_\theta}(z_{t+\tau+1} | z_{t+\tau})\right]$

State-value function: $V_{\pi_\theta}(s_t) = E_{a_t \sim \pi_\theta(\cdot | s_t)}\left[Q_{\pi_\theta}(z_t)\right]$

Advantage function: $A_{\pi_\theta}(z_t) = Q_{\pi_\theta}(z_t) - V_{\pi_\theta}(s_t)$

Expected reward: $J(\theta) = E_{s \sim \rho_0}\left[V_{\pi_\theta}(s)\right]$

Policy Gradient Theorem

$$\nabla_\theta J(\theta) = \int_{\mathcal{Z}} dz \rho^{\pi_\theta}(z) \nabla_\theta \log \pi_\theta(a|s) Q_{\pi_\theta}(z)$$

$$= E_{z \sim \rho^{\pi_\theta}} \left[Q_{\pi_\theta}(z) \nabla_\theta \log \pi_\theta(a|s) \right]$$

where,

$$P_t^{\pi_\theta}(z_t) = \int_{\mathcal{Z}_t} dz_0 \ldots dz_{t-1} P_0^{\pi_\theta}(z_0) \prod_{\tau=1}^{t} P^{\pi_\theta}(z_\tau | z_{\tau-1}), \qquad \rho^{\pi_\theta}(z) = \sum_{t=0}^{\infty} \gamma^t P_t^{\pi_\theta}(z)$$

[Sutton et al., 2000]

Monte-Carlo PG Estimation

Monte-Carlo PG Estimation

$$\begin{aligned}\nabla_\theta J(\theta) &= E_{z\sim\rho^{\pi_\theta}}\left[Q_{\pi_\theta}(z)\nabla_\theta \log \pi_\theta(a|s)\right]\\ &\approx L_\theta^{MC} = \frac{1}{n}\sum_{i=1}^{n}\nabla_\theta \log \pi_\theta(a_i|s_i)Q_{\pi_\theta}(z_i) \quad \forall \quad \{z_i\}_{i=1}^{n} \sim \rho^{\pi_\theta}\end{aligned}$$

where n is the sample size.

Estimating the Q-function

- Monte-Carlo/TD(1) action-value estimates:

$$Q_{\pi_\theta}(z_t) = E\left[\sum_{\tau=0}^{\infty} \gamma^\tau r(z_{t+\tau}) \;\middle|\; z_{t+\tau+1} \sim P^{\pi_\theta}(z_{t+\tau+1}|z_{t+\tau})\right]$$

Expectation over multiple trajectory

$$\approx Q_t^{MC} = \left[\sum_{\tau=0}^{\infty} \gamma^\tau r(z_{t+\tau}) \;\middle|\; z_{t+\tau+1} \sim P^{\pi_\theta}(z_{t+\tau+1}|z_{t+\tau})\right]$$

Single trajectory

- Function approximation for V(s): TD(1)+Advantage (e.g. GAE[1])
- Function approximation for Q(s,a): TD(0), TD(λ)

[1][Schulman et al., 2015]

Monte-Carlo Estimation

Exact gradient **MC approximation**

$$\int_{\mathcal{Z}} dz \rho^{\pi_\theta}(z) \nabla_\theta \log \pi_\theta(a|s) Q_{\pi_\theta}(z) \not\approx \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \log \pi_\theta(a_i|s_i) Q_{\pi_\theta}(z_i)$$

+ Returns **unbiased** policy gradient estimates
+ Computationally efficient, i.e., **scalable**

- **Statistically inefficient** (high sample complexity)
- **Low accuracy**
- **High variance**

[Ilyas et al., 2018]

Matrix Representation of MC-PG

Score function: $\mathbf{u}(z) = \nabla_\theta \log \pi_\theta(a|s)$

For samples $\{z_i\}_{i=1}^n \sim \rho^{\pi_\theta}$,

$$\mathbf{U} = [\mathbf{u}(z_1), \mathbf{u}(z_2), ..., \mathbf{u}(z_n)]$$

$$\mathbf{Q} = [Q_{\pi_\theta}(z_1), Q_{\pi_\theta}(z_2), ..., Q_{\pi_\theta}(z_n)]$$

Monte-Carlo (MC) estimate of policy gradient:

$$\mathbf{L}_\theta^{MC} = \frac{1}{n}\sum_{i=1}^{n} Q_{\pi_\theta}(z_i)\mathbf{u}(z_i) = \frac{1}{n}\mathbf{UQ}$$

Bayesian Quadrature

Bayesian Quadrature

$$\nabla_\theta J(\theta) = \int_{\mathcal{Z}} dz \rho^{\pi_\theta}(z) \nabla_\theta \log \pi_\theta(a|s) Q_{\pi_\theta}(z)$$

Overview: Replace $Q_{\pi_\theta}(z)$ with a function approximation that:

1. closely fits $Q_{\pi_\theta}(z)$ near sampled locations $\{z_i\}_{i=1}^n \sim \rho^{\pi_\theta}$.
2. Offers an analytical solution to the policy gradient integral.

Bayesian Quadrature

$$\nabla_\theta J(\theta) = \int_{\mathcal{Z}} dz \rho^{\pi_\theta}(z) \nabla_\theta \log \pi_\theta(a|s) Q_{\pi_\theta}(z)$$

Step 1: Choose a prior stochastic process over $Q_{\pi_\theta}(z)$.

- common choice is a Gaussian process (GP):

$$\mathbf{Q}_{\pi_\theta} = [Q_{\pi_\theta}(z_1), Q_{\pi_\theta}(z_2), ..., Q_{\pi_\theta}(z_n)]^\top \sim \mathcal{N}(0, K)$$

$$K_{p,q} = k(z_p, z_q)$$

Bayesian Quadrature

Step 2: Conditioning the GP prior on the samples $\{z_i\}_{i=1}^n \sim \rho^{\pi_\theta}$ the posterior moments of $Q_{\pi_\theta}(z)$ are as follows:

$$E\left[Q_{\pi_\theta}(z)|\mathcal{D}\right] = \mathbf{k}(z)^\top(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{Q}$$

$$Cov\left[Q_{\pi_\theta}(z_1), Q_{\pi_\theta}(z_2)|\mathcal{D}\right] = k(z_1, z_2) - \mathbf{k}(z_1)^\top(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{k}(z_2)$$

$$\mathbf{k}(z) = [k(z_1, z), ..., k(z_n, z)],\ \mathbf{K} = [\mathbf{k}(z_1), ..., \mathbf{k}(z_n)]$$

Bayesian Quadrature

Step 3: Use the posterior over integrand to compute policy gradient mean and covariance.

$$L_\theta^{BQ} = E\left[\nabla_\theta J(\theta)|\mathcal{D}\right] = \int_z \rho^{\pi_\theta}(z)u(z)E\left[Q_{\pi_\theta}(z)|\mathcal{D}\right]dz$$

$$= \left(\int_z \rho^{\pi_\theta}(z)u(z)\mathbf{k}(z)^\top dz\right)(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{Q}$$

$$C_\theta^{BQ} = Cov[\nabla_\theta J(\theta)|\mathcal{D}] = \int_{z_1,z_2} \rho^{\pi_\theta}(z_1)\rho^{\pi_\theta}(z_2)u(z_1)Cov[Q_{\pi_\theta}(z_1), Q_{\pi_\theta}(z_2)|\mathcal{D}]u(z_2)^\top dz_1 dz_2$$

$$= \int_{z_1,z_2} \rho^{\pi_\theta}(z_1)\rho^{\pi_\theta}(z_2)u(z_1)\left(k(z_1,z_2) - \mathbf{k}(z_1)^\top(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{k}(z_2)\right)u(z_2)^\top dz_1 dz_2$$

Bayesian Quadrature

Step 3: Use the posterior over integrand to compute policy gradient mean and covariance.

$$L_\theta^{BQ} = E\left[\nabla_\theta J(\theta)|\mathcal{D}\right] = \int_z \rho^{\pi_\theta}(z)u(z)E\left[Q_{\pi_\theta}(z)|\mathcal{D}\right]dz$$

$$= \left(\int_z \rho^{\pi_\theta}(z)u(z)\mathbf{k}(z)^\top dz\right)(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{Q}$$

Appropriate kernel choice provides closed form solution!

$$C_\theta^{BQ} = Cov[\nabla_\theta J(\theta)|\mathcal{D}] = \int_{z_1,z_2} \rho^{\pi_\theta}(z_1)\rho^{\pi_\theta}(z_2)u(z_1)Cov[Q_{\pi_\theta}(z_1), Q_{\pi_\theta}(z_2)|\mathcal{D}]u(z_2)^\top dz_1 dz_2$$

$$= \int_{z_1,z_2} \rho^{\pi_\theta}(z_1)\rho^{\pi_\theta}(z_2)u(z_1)\left(k(z_1,z_2) - \mathbf{k}(z_1)^\top(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{k}(z_2)\right)u(z_2)^\top dz_1 dz_2$$

[Ghavamzadeh and Engel, 2007]

Useful Identities

- Expectation of a score vector under the policy distribution is **0**:

$$E_{a\sim\pi_\theta(.|s)}\left[u(z)\right] = E_{a\sim\pi_\theta(.|s)}\left[\nabla_\theta \log \pi_\theta(a|s)\right] = \int_{\mathcal{A}} \pi_\theta(a|s)\nabla_\theta \log \pi_\theta(a|s)da$$

$$= \int_{\mathcal{A}} \pi_\theta(a|s)\frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)}da = \nabla_\theta \int_{\mathcal{A}} \pi_\theta(a|s)da$$

$$= \nabla_\theta(1) = \boxed{0}$$

- **Fisher Information Matrix (G)**:

$$\mathbf{G} = E_{z\sim\rho^{\pi_\theta}}[\mathbf{u}(z)\mathbf{u}(z)^\top] \approx \frac{1}{n}\mathbf{U}\mathbf{U}^\top$$

Kernel Choice

The kernel choice that solves PG integral in closed form:

$$k(z_1, z_2) = c_1 \underbrace{k_s(s_1, s_2)}_{\text{State Kernel}} + c_2 \underbrace{k_f(z_1, z_2)}_{\text{Fisher Kernel}} \text{ with } k_f(z_1, z_2) = \mathbf{u}(z_1)^\top \mathbf{G}^{-1} \mathbf{u}(z_2),$$

where ***G*** is the **Fisher Information Matrix**.

Matrix representation:

$$\mathbf{k}(z) = c_1 \mathbf{k}_s(s) + c_2 \mathbf{k}_f(z) \quad \Big| \quad \mathbf{k}_s(s) = [k_s(s_1, s), \ldots, k_s(s_n, s)] \quad \Big| \quad \mathbf{k}_f(z) = \mathbf{U}^\top \mathbf{G}^{-1} \mathbf{u}(z)$$

$$\mathbf{K} = c_1 \mathbf{K}_s + c_2 \mathbf{K}_f \quad \Big| \quad \mathbf{K}_s = [\mathbf{k}_s(s_1), \ldots, \mathbf{k}_s(s_n)] \quad \Big| \quad \mathbf{K}_f = \mathbf{U}^\top \mathbf{G}^{-1} \mathbf{U}$$

Why this kernel choice?

The kernel choice that solves PG integral in closed form:

$$k(z_1, z_2) = c_1 \underbrace{k_s(s_1, s_2)}_{\text{State Kernel}} + c_2 \underbrace{k_f(z_1, z_2)}_{\text{Fisher Kernel}} \text{ with } k_f(z_1, z_2) = \mathbf{u}(z_1)^\top \mathbf{G}^{-1} \mathbf{u}(z_2),$$

$$\mathbf{L}_\theta^{BQ} = \left(\int_z \rho^{\pi_\theta}(z) \mathbf{u}(z) \mathbf{k}(z)^\top dz \right) (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{Q} = E_{z \sim \rho^{\pi_\theta}} \left[\mathbf{u}(z) \mathbf{k}(z)^\top \right] \left(\mathbf{K} + \sigma^2 \mathbf{I} \right)^{-1} \mathbf{Q}$$

$$= \left(\underbrace{c_1 E_{z \sim \rho^{\pi_\theta}} \left[\mathbf{u}(z) \mathbf{k_s}(s)^\top \right]}_{\boxed{0}} + c_2 E_{z \sim \rho^{\pi_\theta}} \left[\mathbf{u}(z) \mathbf{k_f}(z)^\top \right] \right) \left(\mathbf{K} + \sigma^2 \mathbf{I} \right)^{-1} \mathbf{Q}$$

$$= c_2 E_{z \sim \rho^{\pi_\theta}} \left[\mathbf{u}(z) \mathbf{u}(z)^\top \right] \mathbf{G}^{-1} \mathbf{U} \left(\mathbf{K} + \sigma^2 \mathbf{I} \right)^{-1} \mathbf{Q}$$

$$= \mathbf{U} \left(\mathbf{K} + \sigma^2 \mathbf{I} \right)^{-1} \mathbf{Q}$$

BQ-PG Posterior Moments

For samples $\{z_i\}_{i=1}^{n} \sim \rho^{\pi_\theta}$ and score function $\mathbf{u}(z) = \nabla_\theta \log \pi_\theta(a|s)$,

$$\mathbf{U} = [\mathbf{u}(z_1), \mathbf{u}(z_2), \ldots, \mathbf{u}(z_n)] \qquad \mathbf{Q}^{MC} = [Q_{\pi_\theta}(z_1), Q_{\pi_\theta}(z_2), \ldots, Q_{\pi_\theta}(z_n)]$$

$$\mathbf{L}_\theta^{BQ} = c_2 \mathbf{U}(c_1 \mathbf{K}_s + c_2 \mathbf{K}_f + \sigma^2 \mathbf{I})^{-1} \mathbf{Q}$$

Policy Gradient Mean

$$\mathbf{C}^{BQ} = c_2 \mathbf{G} - c_2^2 \mathbf{U} \left(c_1 \mathbf{K}_s + c_2 \mathbf{K}_f + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{U}^\top$$

Policy Gradient Covariance

More intuition behind this kernel choice

$$E\left[Q_{\pi_\theta}(z)|\mathcal{D}\right] = \mathbf{k}(z)^\top(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{Q}$$
$$Cov\left[Q_{\pi_\theta}(z_1), Q_{\pi_\theta}(z_2)|\mathcal{D}\right] = k(z_1, z_2) - \mathbf{k}(z_1)^\top(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{k}(z_2)$$

Action Value Posterior

$$E\left[V_{\pi_\theta}(s)|\mathcal{D}\right] = \boxed{c_1}\mathbf{k}_s(s)^\top(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{Q}$$
$$Cov\left[V_{\pi_\theta}(s_1), V_{\pi_\theta}(s_2)|\mathcal{D}\right] = \boxed{c_1}k_s(s_1, s_2) - \boxed{c_1^2}\mathbf{k}_s(s_1)^\top(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{k}_s(s_2)$$

State Value Posterior

$$E\left[A_{\pi_\theta}(z)|\mathcal{D}\right] = \boxed{c_2}\mathbf{k}_f(z)^\top(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{Q}$$
$$Cov\left[A_{\pi_\theta}(z_1), A_{\pi_\theta}(z_2)|\mathcal{D}\right] = \boxed{c_2}k_f(z_1, z_2) - \boxed{c_2^2}\mathbf{k}_f(z_1)^\top(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{k}_f(z_2)$$

Advantage Value Posterior

MC-PG vs BQ-PG

Monte-Carlo estimation of policy gradient:

$$\mathbf{L}_\theta^{MC} = \frac{1}{n}\mathbf{U}\mathbf{Q}$$

Bayesian Quadrature estimation of policy gradient:

$$\mathbf{L}_\theta^{BQ} = \boxed{c_2}\mathbf{U}(c_1\mathbf{K}_s + c_2\mathbf{K}_f + \sigma^2\mathbf{I})^{-1}\mathbf{Q}$$

Limiting Cases of BQ-PG

When $c_1 = 0$:

$$\mathbf{L}_\theta^{BQ}\Big|_{c_1=0} = \frac{c_2}{\sigma^2 + c_2 n}\mathbf{U}\mathbf{Q} \propto \mathbf{L}_\theta^{MC}$$

$$\mathbf{C}_\theta^{BQ}\Big|_{c_1=0} = \frac{\sigma^2 c_2}{\sigma^2 + c_2 n}\mathbf{G} \propto c_2\mathbf{G}$$

Highlights:

1. BQ-PG's posterior mean reduces to **MC-PG**.
2. BQ-PG's posterior covariance is a scalar multiple of the prior covariance/**F.I.M** (***G***).

When $c_2 = 0$:

$$\mathbf{L}_\theta^{BQ}\Big|_{c_2=0} = 0 \qquad \mathbf{C}_\theta^{BQ}\Big|_{c_2=0} = 0$$

Highlights:

1. Posterior moments of the policy gradient vanish upon removing the Fisher kernel.

Computational Analysis

Computational Complexity of BQ-PG

$$\mathbf{L}_\theta^{BQ} = c_2 \mathbf{U}(c_1 \mathbf{K}_s + c_2 \mathbf{K}_f + \sigma^2 \mathbf{I})^{-1} \mathbf{Q}$$

$$\underbrace{(c_1 \mathbf{K}_s + c_2 \mathbf{K}_f + \sigma^2 \mathbf{I})^{-1} \mathbf{Q}}$$

Inverse Matrix Vector Multiplication (iMVM)
(Cubic time complexity => **Bottleneck**)

$$\underbrace{c_2 \mathbf{U} \, \_}$$

Vector-Jacobian Product (vJP)
(1 Autodiff backward operation => **Efficient**)

$$\mathbf{L}_\theta^{BQ}$$

Efficient iMVM Implementation

$$(c_1\mathbf{K}_s + c_2\mathbf{K}_f + \sigma^2\mathbf{I})^{-1}\mathbf{Q}$$

Naive Matrix Inversion

- Cubic time complexity
- Quadratic space complexity

Does not scale to high-dimensional settings

Conjugate Gradient for iMVM

Given a Matrix-Vector-Multiplication (MVM) function with time complexity $O(\mathcal{M})$:

- Time complexity: $O(p*\mathcal{M})$
- p: Number of CG iterations ($p << n$)

Naive MVM

- Quadratic time and space complexity

Does not scale to high-dimensional settings

Efficient MVM

- Linear time and space complexity

Scales to high-dimensional settings

Efficient MVM Implementation

$$\underbrace{(c_1\mathbf{K}_s + c_2\mathbf{K}_f + \sigma^2\mathbf{I})^{-1}\mathbf{Q}}$$

$$\mathbf{K}_s\mathbf{Q}$$

Since state kernel is arbitrary, efficient MVM requires a **general** interpolation strategy:

- *Structured Kernel Interpolation (SKI)*
  - Scales **linearly**.
  - **Additional scalability** for special kernel families.

$$\mathbf{K}_f\mathbf{Q}$$

Special structure of $\boldsymbol{K}_f$ enables for efficient MVM through autodiff backward calls:

I. Vector-Jacobian Product (vJp)
II. inverse-Hessian-Vector Product
III. Jacobian-Vector Product (Jvp)

- FastSVD for **additional speedup**.

Structured Kernel Interpolation (SKI)

Inducing point approximation: $\underbrace{\mathbf{K}_s}_{\boxed{\text{n x n}}} \approx \underbrace{\mathbf{W}}_{\boxed{\text{n x m}}}\underbrace{\mathbf{K}_s^m}_{\boxed{\text{m x m}}}\underbrace{\mathbf{W}^\top}_{\boxed{\text{m x n}}}$

Interpolation Matrix

Using a sparse interpolation matrix ***W***:

- Bicubic interpolation, i.e., 4 non-zero elements per row.

$$\mathbf{K}_s\mathbf{Q} \approx \mathbf{W}\left(\mathbf{K}_s^m\left(\mathbf{W}^\top\mathbf{Q}\right)\right)$$

Structured Kernel Interpolation (SKI)

- **Kronecker method**:
  - Product kernel
  - Inducing points on a multidimensional grid
- **Toeplitz method**:
  - stationary kernel
  - Inducing points on a 1D grid.

Complexity	**SKI**	**SKI + Kronecker**	**SKI + Topelitz**
Time	$O(n+m^2)$	$O(n+Ym^{1+1/Y})$	$O(n+m*log(m))$
Space	$O(n+m^2)$	$O(n+Ym^{2/Y})$	$O(n+m)$

Fisher Kernel MVM using only AutoDiff

$$\mathbf{K}_f\mathbf{v} = \left(\mathbf{U}^\top\left(\mathbf{G}^{-1}\left(\mathbf{U}\mathbf{v}\right)\right)\right) = \left(\underbrace{\frac{\partial \mathcal{L}}{\partial \theta}}_{\text{Jvp}}\left(\underbrace{\mathbf{G}^{-1}}_{\text{iHvp}}\left(\underbrace{\left(\frac{\partial \mathcal{L}}{\partial \theta}\right)^\top \mathbf{v}}_{\text{vJp}}\right)\right)\right)$$

Too many backward calls !!

where $\mathcal{L} = [\log \pi_\theta(a_1|s_1), ..., \log \pi_\theta(a_n|s_n)]$

Complexity in terms of reverse-mode automatic differentiation (AD):

1. **vJp**: 1 backward pass
2. **Hvp**: 2 backward passes
3. **iHvp**: 2**p* backward passes (*p*: Number of CG iterations)
4. **Jvp**: 2 backward passes (or 1 forward pass in forward-mode AD)

Fisher Kernel MVM using SVD (Faster!)

Let $\mathbf{U} = \mathbf{P}\mathbf{\Lambda}\mathbf{R}^\top$ (SVD),

then $\mathbf{G} = \frac{1}{n}\mathbf{U}\mathbf{U}^\top = \frac{1}{n}\mathbf{P}\mathbf{\Lambda}^2\mathbf{P}^\top$ and,

$$\mathbf{K}_f = \mathbf{U}^\top\mathbf{G}^{-1}\mathbf{U} = n\mathbf{R}\mathbf{\Lambda}\mathbf{P}^\top\left(\mathbf{P}\mathbf{\Lambda}^{-2}\mathbf{P}^\top\right)\mathbf{P}\mathbf{\Lambda}\mathbf{R}^\top = n\mathbf{R}\mathbf{R}^\top$$

Equivalent to a linear kernel in ***R***!!

- **Randomized SVD**: Fast, scalable and supports implicit MVM !!
    - Linear time MVM ***O(n*δ)***, where *δ* is the rank of ***truncated*** SVD.

Deep BQ-PG (DBQPG)
Scaling BQ-PG to high-dimensional settings

Scaling to High-Dimensional Settings: DBQPG

Linear scaling algorithm:

State kernel MVM (CG inner-loop):
- Deep RBF kernel+kernel learning (**GPU**)
- Kernel Interpolation + Toeplitz method

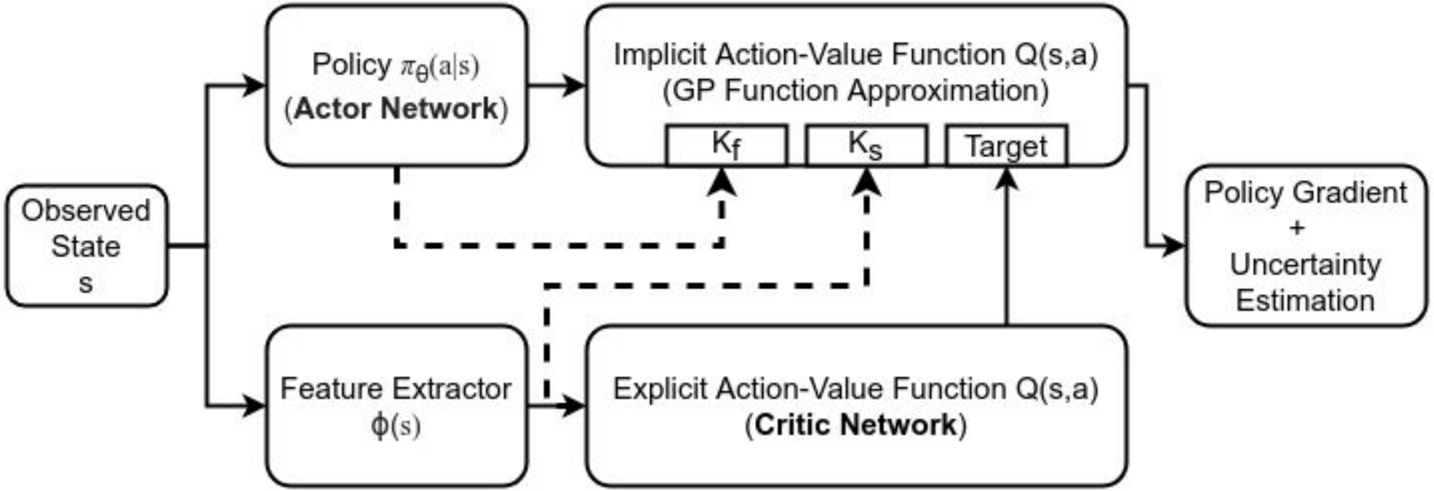
Inverse MVM:
- Conjugate gradient (CG)

$$\mathbf{L}_\theta^{BQ} = c_2 \boxed{\mathbf{U}} (c_1 \boxed{\mathbf{K}_s} + c_2 \boxed{\mathbf{K}_f} + \sigma^2 \mathbf{I})^{\boxed{-1}} \mathbf{Q}$$

Vector-Jacobian Product (vJp):
- 1 backward pass

Fisher kernel MVM (CG inner-loop):
- Randomized SVD → Linear kernel (fast)

DBQPG Algorithm

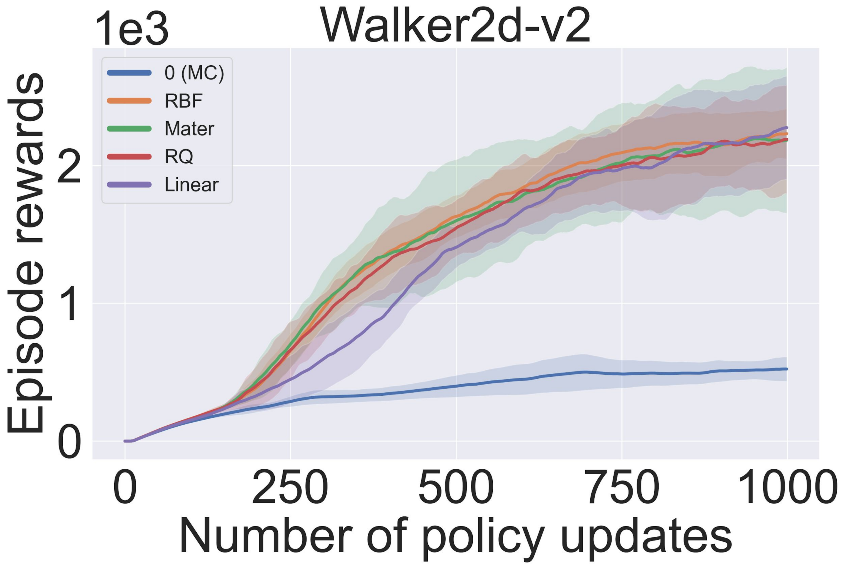


Kernel Variations in DBQPG

Kernel composition $\mathbf{k}(z) = c_1\mathbf{k}_s(s) + c_2\mathbf{k}_f(z)$:

- Fisher kernel (fixed; essential for solving policy gradient integral)
- State kernel (arbitrary; derivation holds for any valid kernel)
  - Base kernels:
    - **RBF**, Matern, Polynomial kernel, etc.
  - Enhancing expressivity of base kernels:
    - **Deep kernels**
      - NN feature extractor + base kernel
    - **Kernel learning**
      - Optimize kernel hyperparameters for GP's MLL

DBQPG State Kernel Selection
(Base kernel comparison)



k_s = *0* (*i.e.*, BQ-PG → MC-PG)

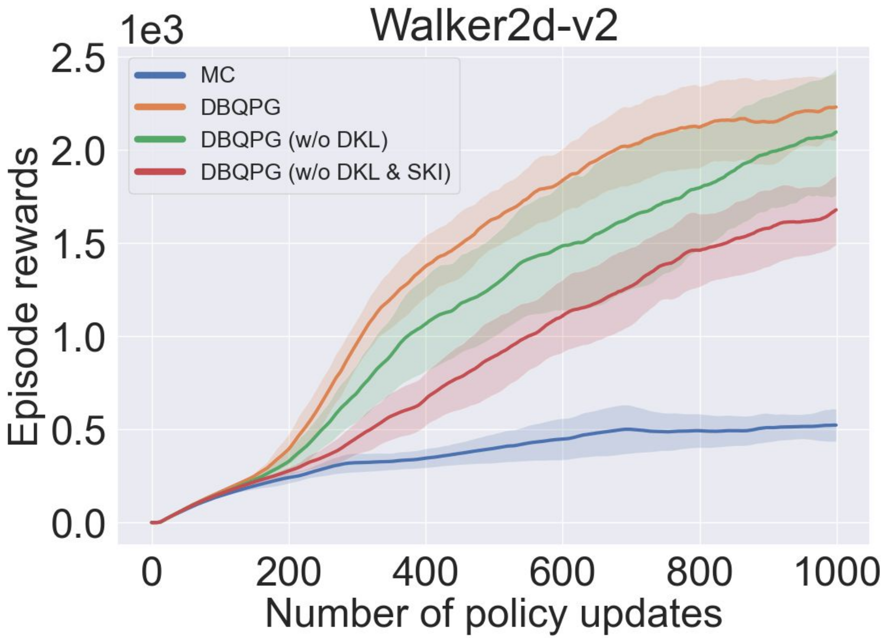
- Bad prior.
- State-value function suppressed to 0.

k_s ≠ *0* (*i.e.*, BQ-PG ↛ MC-PG)

- Doesn't have to be better than MC-PG.
- Yet, most base kernels outperform MC!
  - Even Linear kernel (non-stationary)

k_s = *0* (equivalently MC) results in degeneracy of BQ's performance.

DBQPG Ablation Study
(Role of SKI & DKL)



DBQPG (w/o DKL):-

- Plain RBF kernel (w/o NN bases).

DBQPG (w/o DKL & SKI):-

- Plain RBF kernel (w/o NN bases).
- Replaced SKI with traditional inducing points method.

Deep Kernels and ***SKI*** are both important for superior performance of DBQPG.

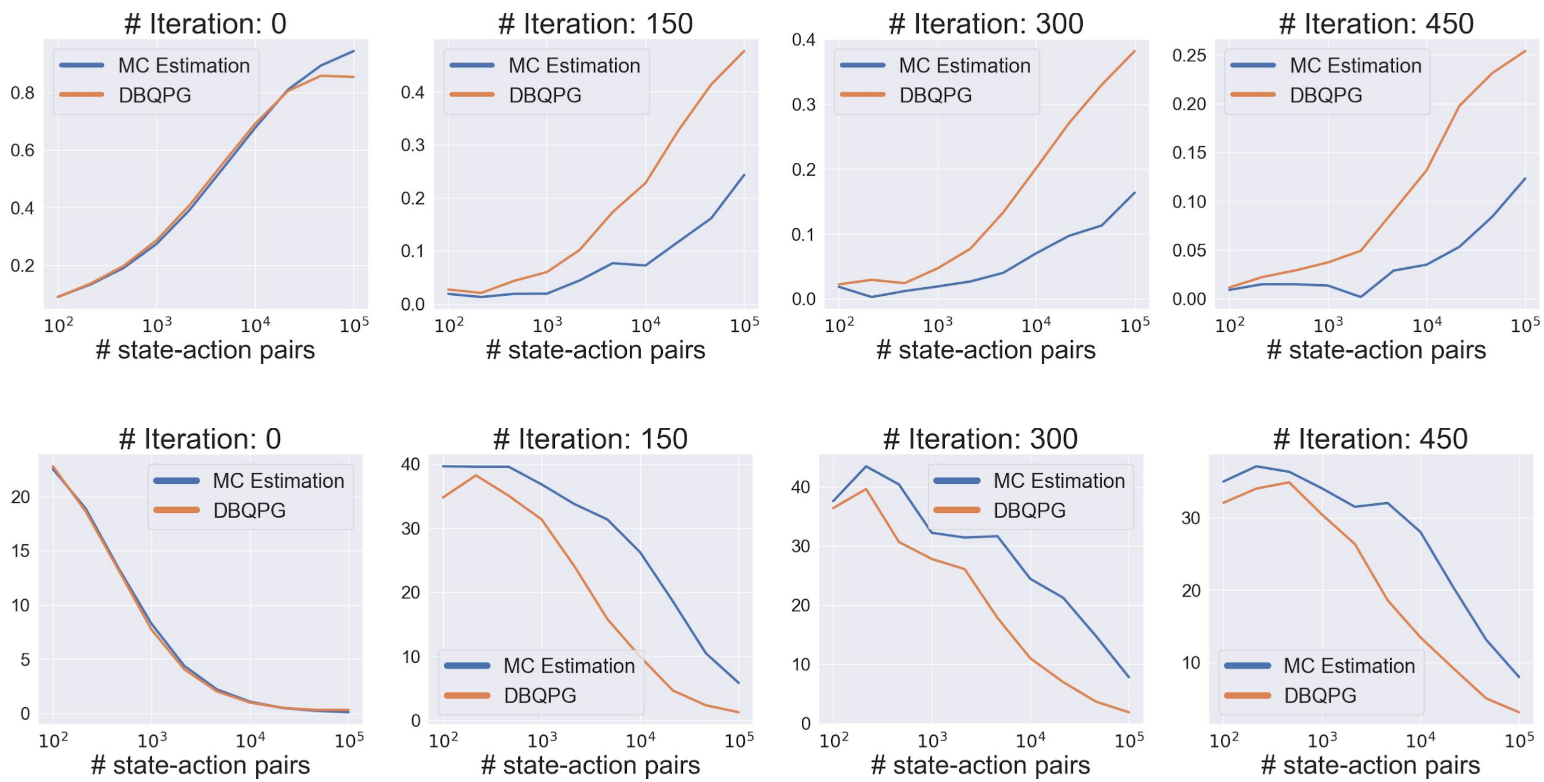
DBQPG vs MC

Gradient Accuracy
(Cosine Similarity)

DBQPG > MC

Gradient Variance
(Normalized)

MC > DBQPG



Summary of DBQPG

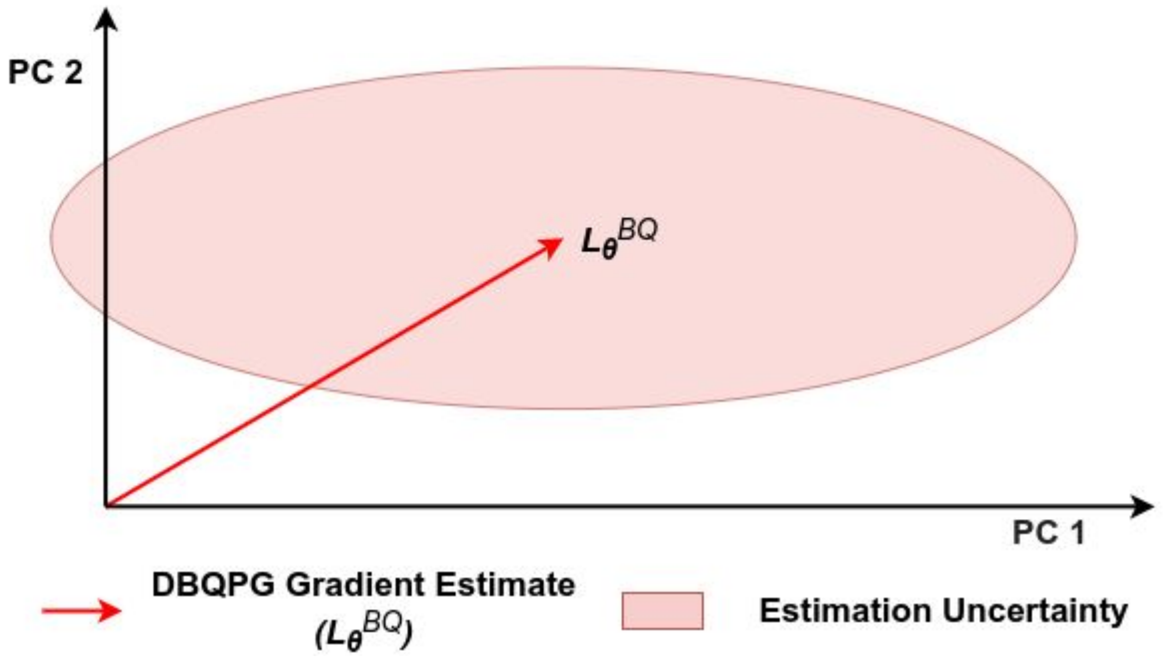
A policy gradient estimator that provides:

1. More **accurate** gradient estimates
2. Lesser **variance** in gradient estimates
3. **Uncertainty** in policy gradient estimation

Can **estimation uncertainty** be used to further improve policy updates? → UAPG

Uncertainty Aware Policy Gradient (UAPG)

Uncertainty Aware Policy Gradient



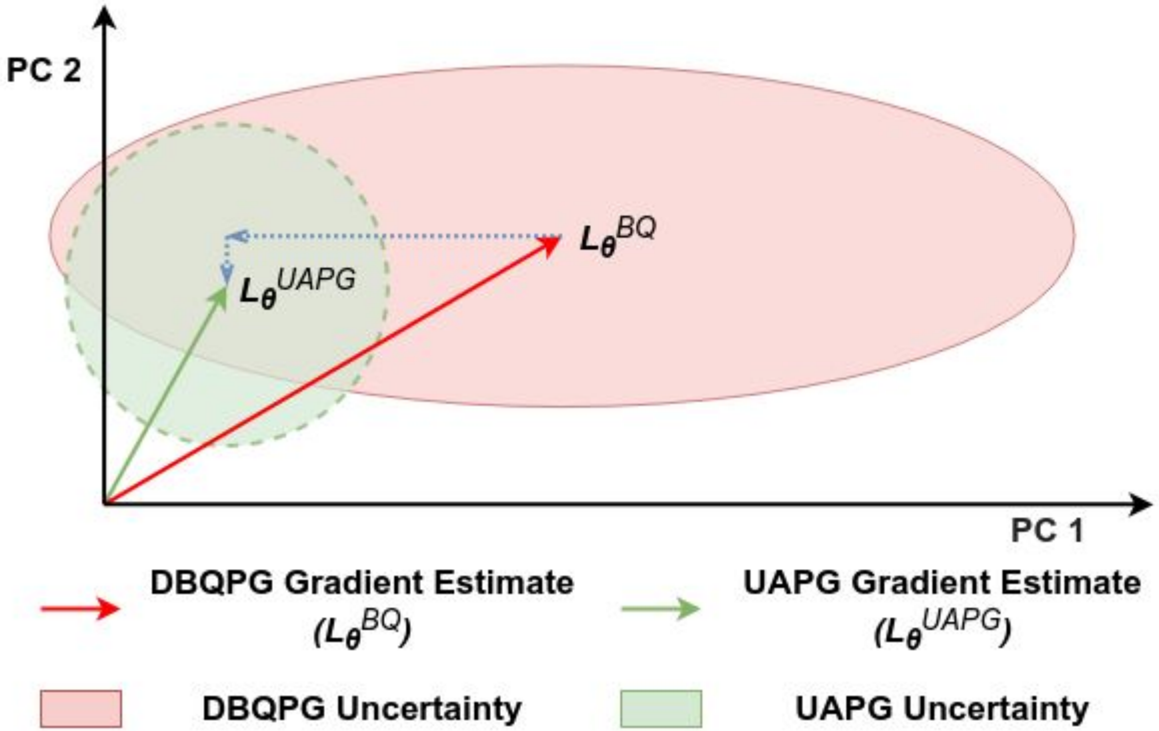
DBQPG update:

- Uses the same learning rate for all gradient components, thus neglecting their respective uncertainties.
- Greater uncertainty increases the risk of large policy updates.

UAPG step-size adjustment:

- Offers a policy update with uniform uncertainty in all the component directions.
- Covariance is **identity matrix**.

Uncertainty Aware Policy Gradient



$$\mathbf{L}_\theta^{UAPG} = \left(\mathbf{C}_\theta^{BQ}\right)^{-\frac{1}{2}} \mathbf{L}_\theta^{BQ}$$

Covariance of UAPG is the identity matrix.

Practical UAPG Algorithm

Randomized (truncated) SVD:

$$\mathbf{C}_\theta^{BQ} \approx \nu_\delta \mathbf{I} + \sum_{i=1}^{\delta} h_i(\nu_i - \nu_\delta)h_i^\top$$

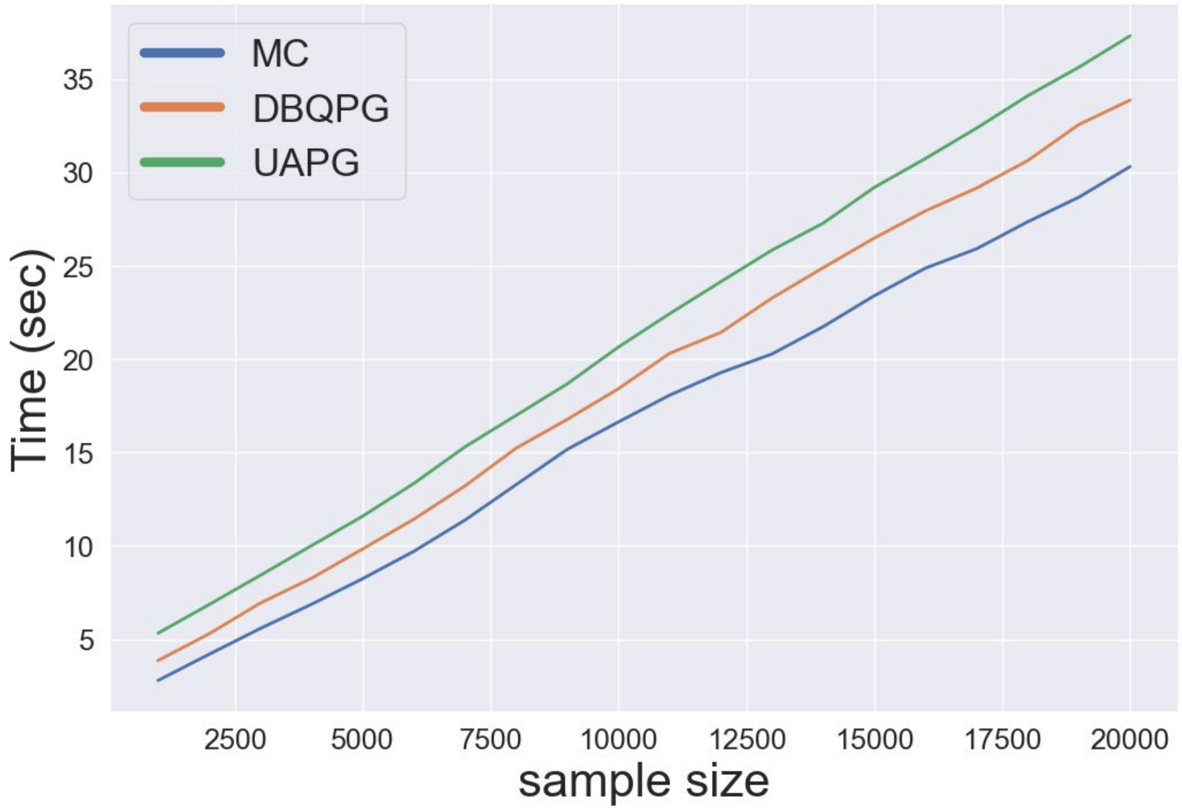
$$\left(\mathbf{C}_\theta^{BQ}\right)^{-\frac{1}{2}} \approx \nu_\delta^{-\frac{1}{2}} \left(\mathbf{I} + \sum\nolimits_{i=1}^{\delta} h_i \left(\sqrt{\nu_\delta/\nu_i} - \mathbf{I}\right) h_i^\top\right)$$

UAPG estimate:

$$\mathbf{L}_\theta^{UAPG} = \left(\mathbf{C}_\theta^{BQ}\right)^{-\frac{1}{2}} \mathbf{L}_\theta^{BQ} \approx \nu_\delta^{-\frac{1}{2}} \left(\mathbf{I} + \sum\nolimits_{i=1}^{\delta} h_i \left(\sqrt{\nu_\delta/\nu_i} - \mathbf{I}\right) h_i^\top\right) \mathbf{L}_\theta^{BQ}$$

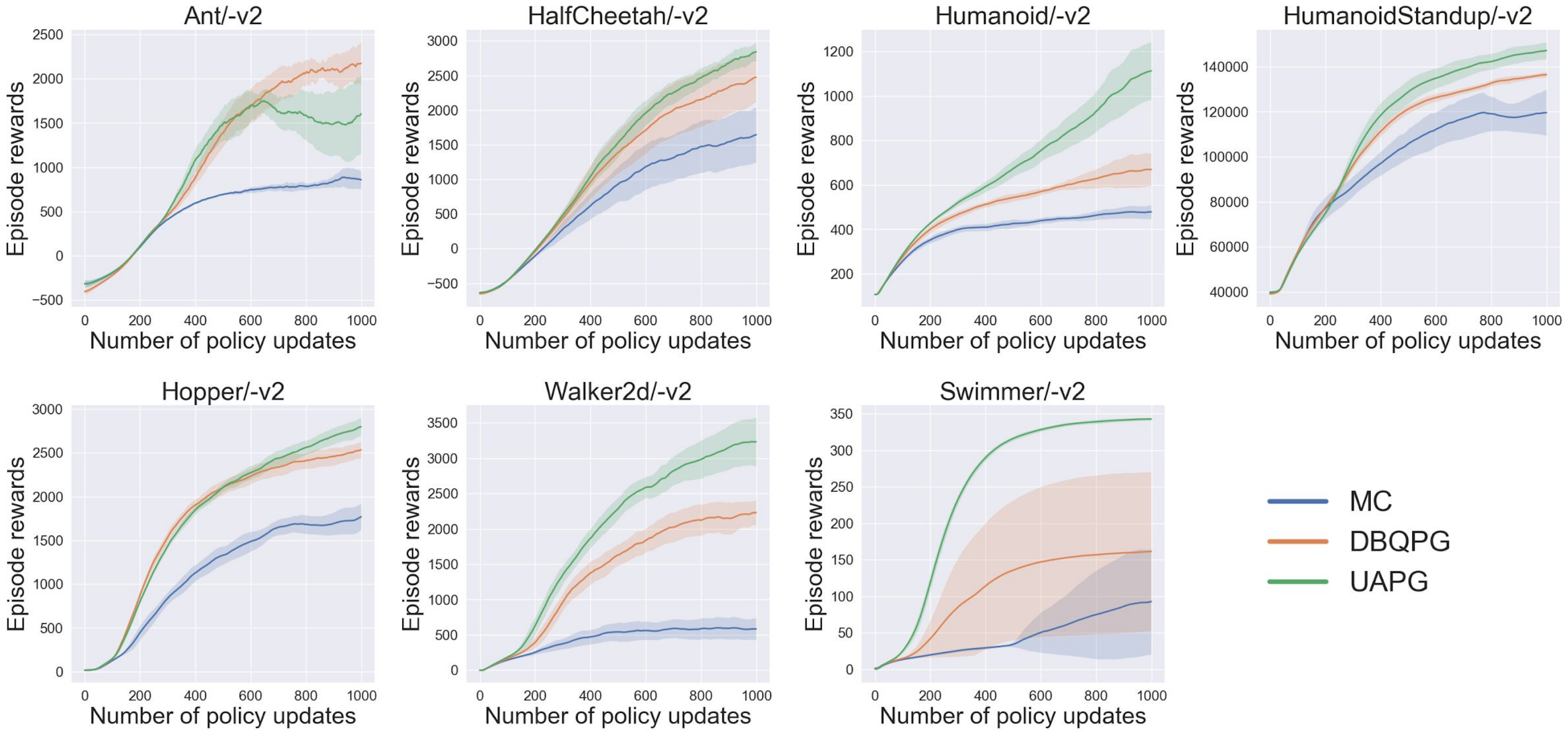
Empirical Analysis

Wall-Clock Time Comparison

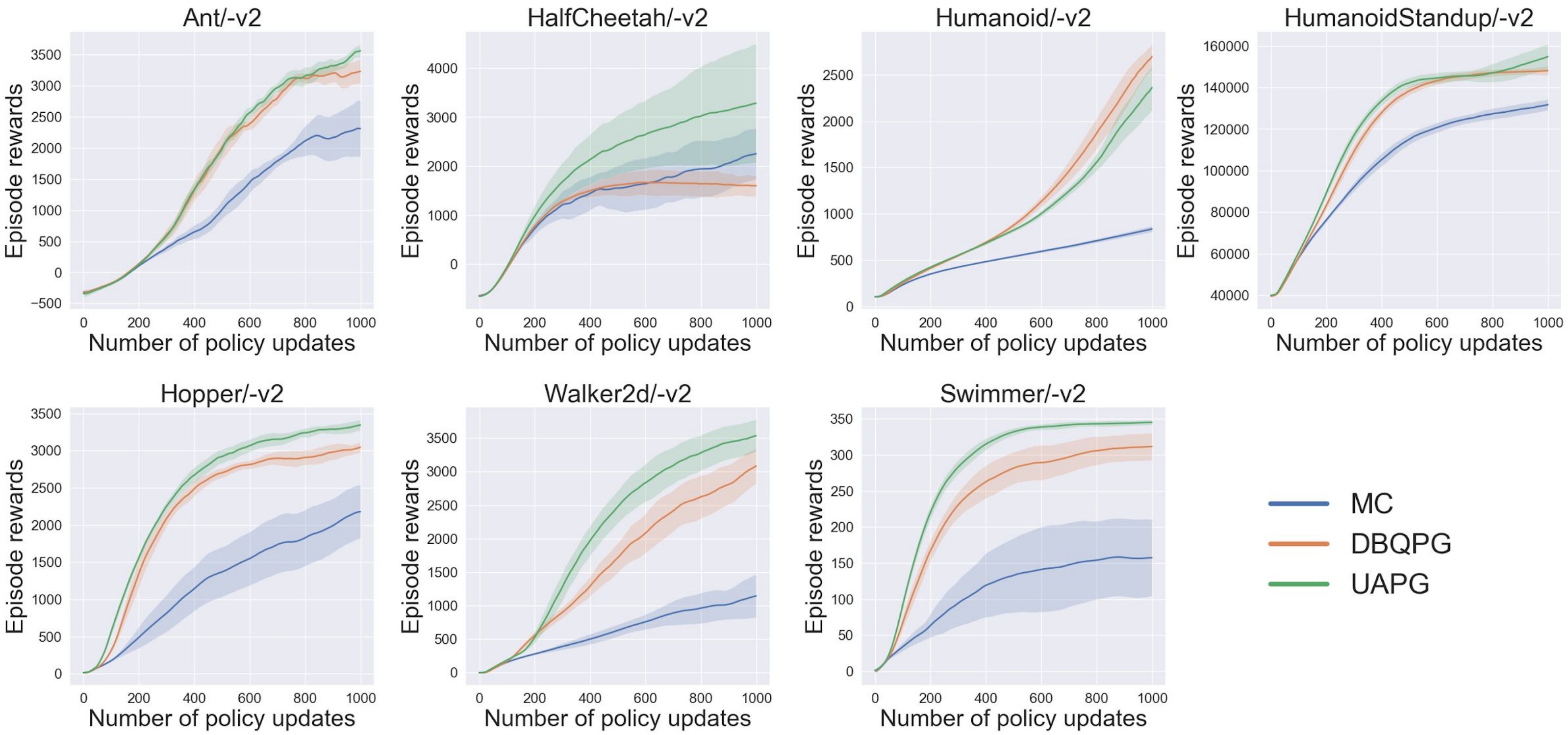


DBQPG and UAPG are linear-scaling methods with negligible overhead over MC.

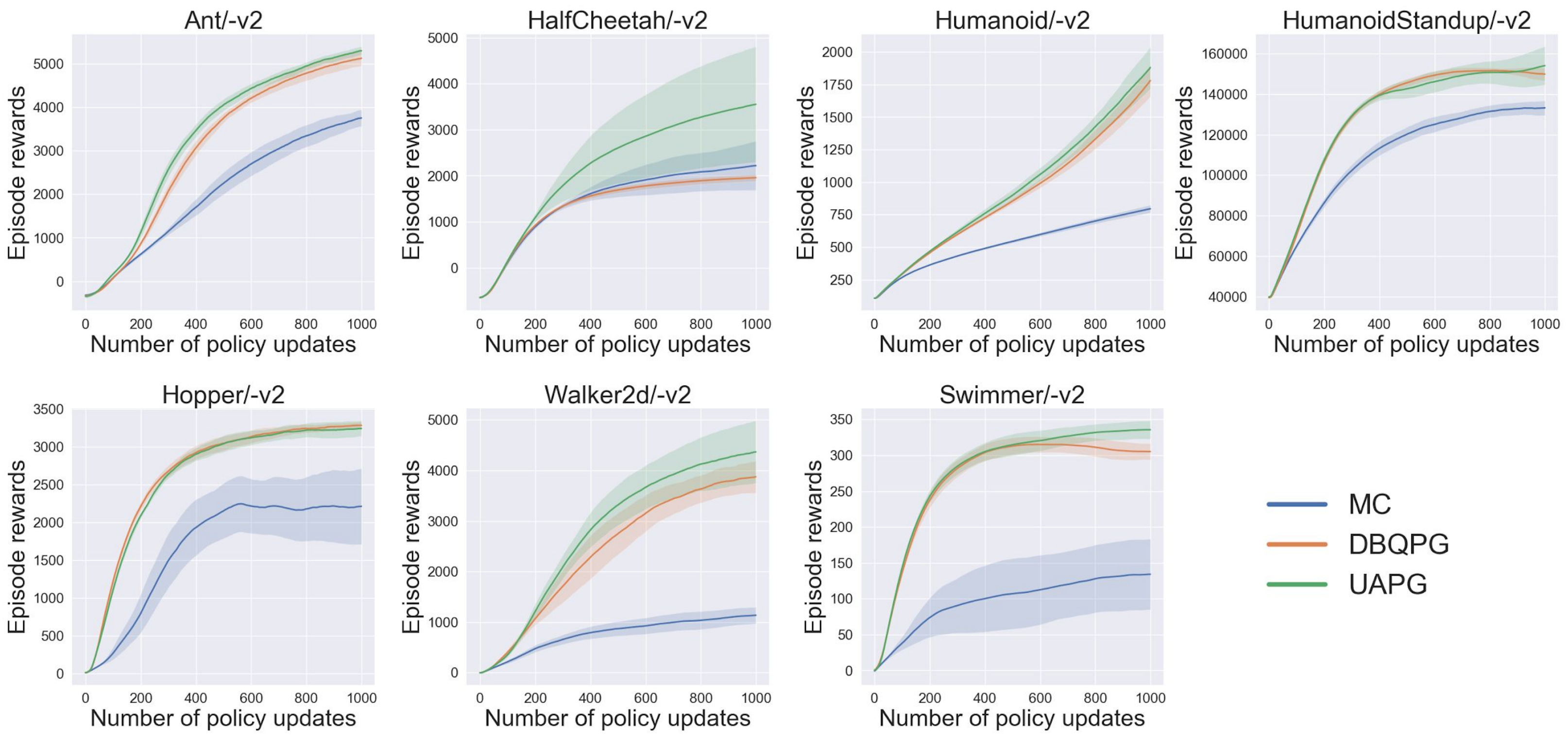
Vanilla Policy Gradient



Natural Policy Gradient (NPG)



Trust Region Policy Optimization (TRPO)



Summary

➢ Deep Bayesian Quadrature Policy Gradient (**DBQPG**)

→ Estimating policy gradients ***more accurately*** with ***fewer samples***.
→ Estimating the ***uncertainty*** in ***stochastic*** gradient estimates.

➢ Uncertainty Aware Policy Gradient (**UAPG**)

→ ***Reliable*** policy updates, i.e., ***adjust*** step-size ↓ using the ***uncertainty*** ↑.

TL; DR: DBQPG and UAPG are statistically efficient alternatives to Monte-Carlo methods that conveniently ***scale*** (linearly) to high-dimensional settings.

Other resources

- Preprint:
  https://arxiv.org/pdf/2006.15637.pdf
- Project website:
  https://akella17.github.io/publications/Deep-Bayesian-Quadrature-Policy-Optimization/
- Blog:
  https://akella17.github.io/blogs/Bayesian-Quadrature-for-Policy-Gradient/
- Source code:
  https://github.com/Akella17/Deep-Bayesian-Quadrature-Policy-Optimization
- Bibtex:

```
@article{ravi2020DBQPG,
title={Deep Bayesian Quadrature Policy Optimization},
author={Akella Ravi Tej and Kamyar Azizzadenesheli and Mohammad Ghavamzadeh
and Anima Anandkumar and Yisong Yue},
journal={arXiv preprint arXiv:2006.15637},
year={2020}
}
```