

Approximate Dynamic Programming with Gaussian Processes

Marc Deisenroth



American Control Conference
Seattle, WA, USA

June 13, 2008

joint work with Jan Peters¹ and Carl Edward Rasmussen^{1,2}

¹ Max Planck Institute for Biological Cybernetics, Tübingen, Germany

² Department of Engineering, University of Cambridge, Cambridge, UK

Problem Statement and Objective

- find optimal policy that minimizes expected long-term cost

$$\sum_{k=0}^N \mathbb{E}[g(\mathbf{x}_k, \mathbf{u}_k)]$$

- efficient method to solve this optimal control problem:
dynamic programming (DP)

- ▶ make DP flexible in continuous-valued state and control spaces
- ▶ proper treatment of uncertainties
- ▶ global optimal policy (not a single trajectory)

Approximate Dynamic Programming

problems with DP:

- only tractable for finite number of states and controls
- curse of dimensionality

→ **function approximators** for continuous-valued states

see e.g. (Bertsekas and Tsitsiklis, 1996; Ormoneit and Šen, 2002; Riedmiller, 2005)

limitations of common function approximators:

- parametric (fix function class **before** having observed any data)
- don't model uncertainty

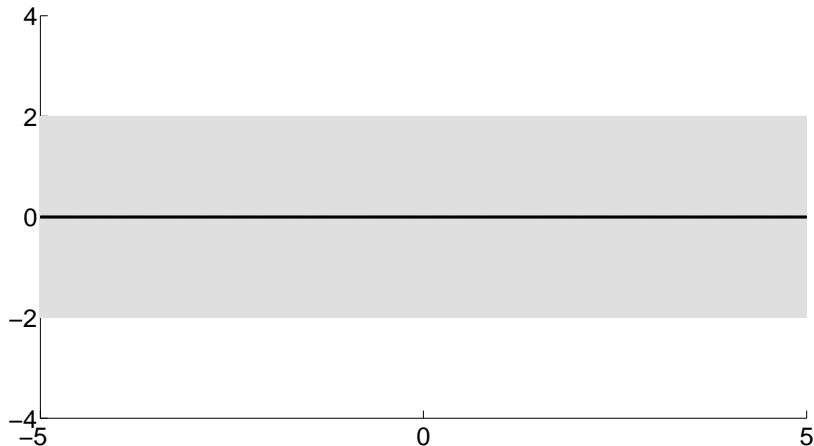
here: use **Gaussian process (GP) regression** (nonparametric, Bayesian)
aka *kriging*, see e.g. (Hernandez Moreno and Grover Gallivan, ACC 2008)

Regression with Gaussian Processes

example: model a function $f : \mathbb{R} \rightarrow \mathbb{R}$ based on noisy observations $f(\mathcal{X}) + \varepsilon$ measured at a set of support points \mathcal{X}

Regression with Gaussian Processes

example: model a function $f : \mathbb{R} \rightarrow \mathbb{R}$ based on noisy observations $f(\mathcal{X}) + \varepsilon$ measured at a set of support points \mathcal{X}

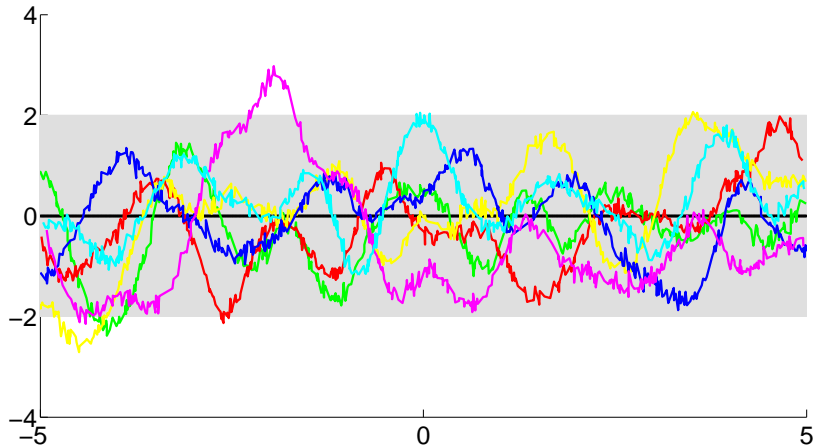


shaded area:

uncertainty about underlying function (twice standard deviation)

Regression with Gaussian Processes

example: model a function $f : \mathbb{R} \rightarrow \mathbb{R}$ based on noisy observations $f(\mathcal{X}) + \varepsilon$ measured at a set of support points \mathcal{X}

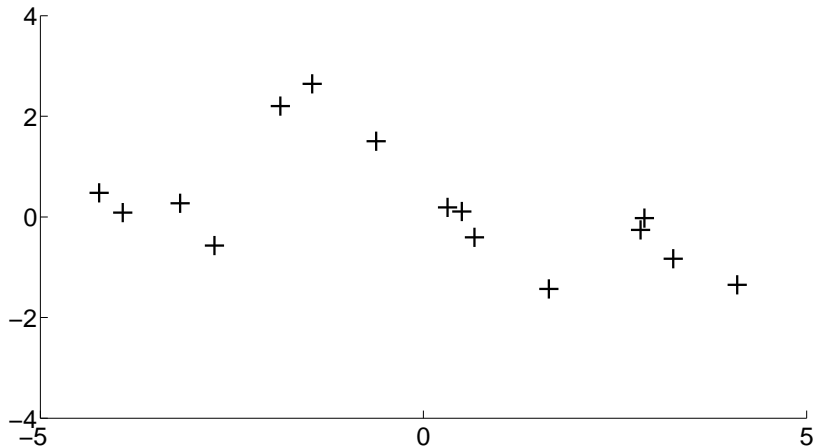


shaded area:

uncertainty about underlying function (twice standard deviation)

Regression with Gaussian Processes

example: model a function $f : \mathbb{R} \rightarrow \mathbb{R}$ based on noisy observations $f(\mathcal{X}) + \varepsilon$ measured at a set of support points \mathcal{X}

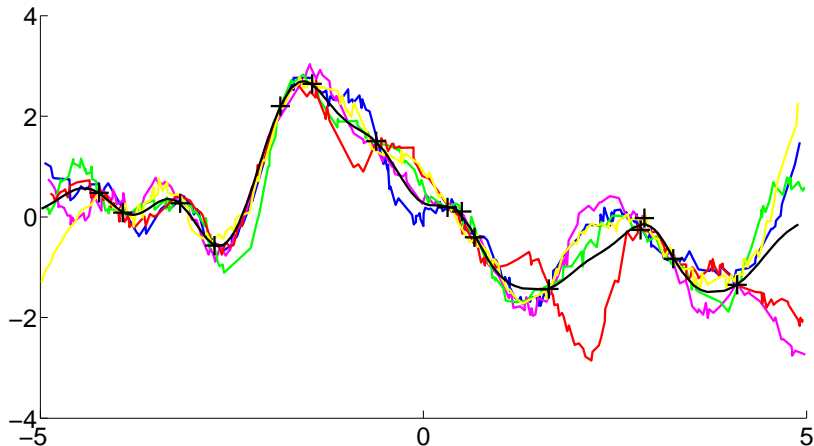


shaded area:

uncertainty about underlying function (twice standard deviation)

Regression with Gaussian Processes

example: model a function $f : \mathbb{R} \rightarrow \mathbb{R}$ based on noisy observations $f(\mathcal{X}) + \varepsilon$ measured at a set of support points \mathcal{X}

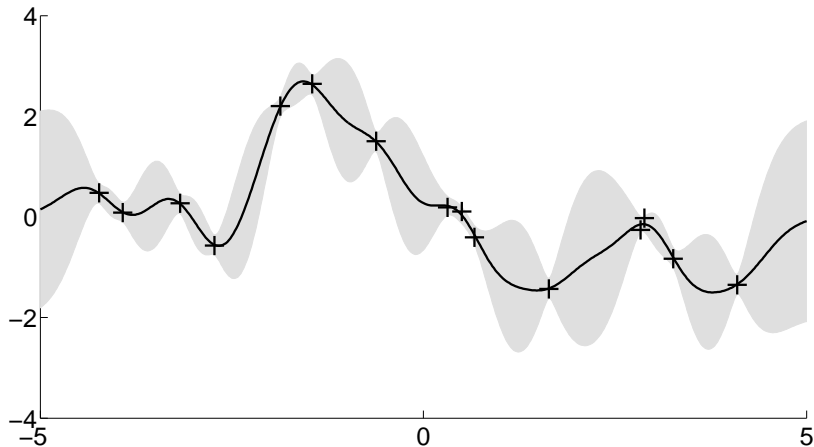


shaded area:

uncertainty about underlying function (twice standard deviation)

Regression with Gaussian Processes

example: model a function $f : \mathbb{R} \rightarrow \mathbb{R}$ based on noisy observations $f(\mathcal{X}) + \varepsilon$ measured at a set of support points \mathcal{X}



shaded area:

uncertainty about underlying function (twice standard deviation)

Objective and Novel Approach

optimal control

- ▶ **continuous** state and action domains
- ▶ model of optimal policy **everywhere**

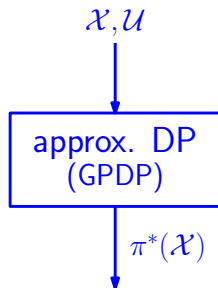
Objective and Novel Approach

optimal control

- ▶ **continuous** state and action domains
- ▶ model of optimal policy **everywhere**

1 Gaussian Process Dynamic Programming

- iteratively model value functions by means of GPs (\mathcal{X}, \mathcal{U} are the support points for the GPs)
- generalization to continuous domains
- get optimal set of optimal state-feedback $\pi^*(\mathcal{X})$



Objective and Novel Approach

optimal control

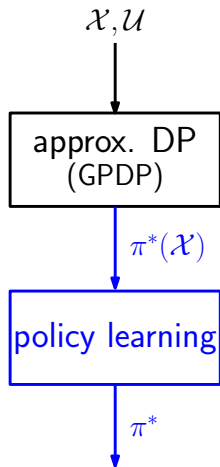
- ▶ **continuous** state and action domains
- ▶ model of optimal policy **everywhere**

1 Gaussian Process Dynamic Programming

- iteratively model value functions by means of GPs (\mathcal{X}, \mathcal{U} are the support points for the GPs)
- generalization to continuous domains
- get optimal set of optimal state-feedback $\pi^*(\mathcal{X})$

2 Policy Learning

- find optimal controls for **any** state based on “noisy observations” $\pi^*(\mathcal{X})$ at \mathcal{X}

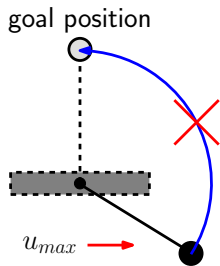


Example System

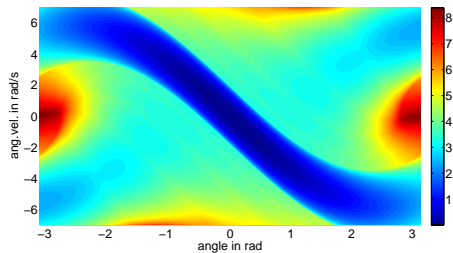
underpowered pendulum swing-up task

challenges:

- **nonlinear** dynamics and controller
- **continuous** states and controls (no spatial discretization)
- interested in closed-loop **policy everywhere**, not a single trajectory



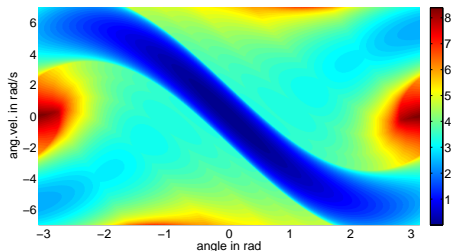
Value Functions



optimal value function

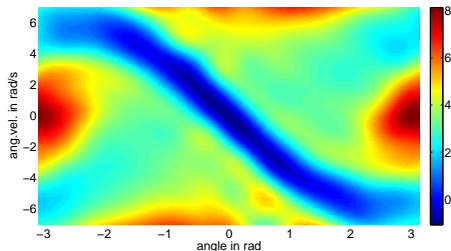
DP with discretization of state
and control spaces
(7.5×10^7 states)

Value Functions



optimal value function

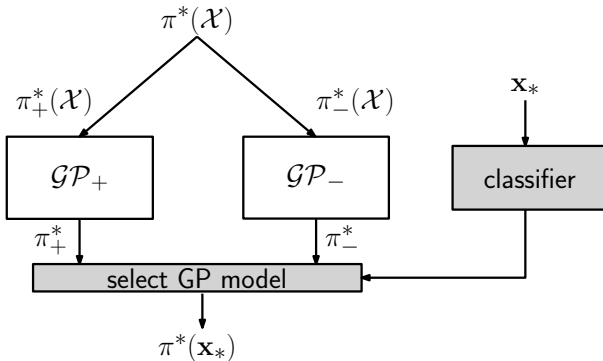
DP with discretization of state and control spaces
(7.5×10^7 states)



mean of value function model
(GPDP):

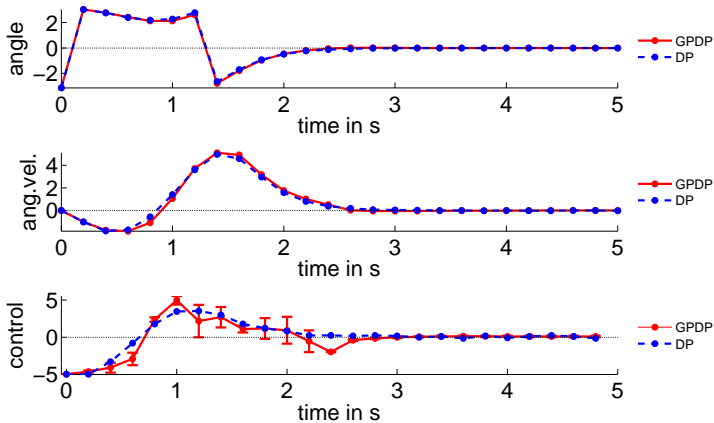
given random input locations \mathcal{X} , determine model of value function V everywhere based on "observations" $V(\mathcal{X})$ at \mathcal{X}
 → return $\pi^*(\mathcal{X})$

Learning a Discontinuous Policy



- state-feedback $\pi^*(\mathcal{X})$ is returned by GPDP
- switch between locally trained policy GPs to model discontinuous policy
- extend finite set of optimal actions to entire state space
 ➔ get model of policy π^*
- similar to mixture-of-experts setting (Jacobs et al., 1991)

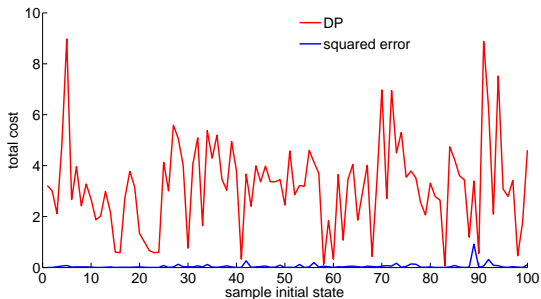
Swing-up Trajectories



→ demo

Some Global Results

- global performance:
 $\text{NMSE}(\text{GPDP}, \text{DP}) \approx 0.06$



- complexity

- standard DP: $\mathcal{O}(|\mathcal{X}_{\text{DP}}|^2 \times |\mathcal{U}|)$,
 $|\mathcal{X}_{\text{DP}}| \approx 6,000$ (regular grid) $\rightarrow 3.6 \times 10^9$ operations
- GPDP: $\mathcal{O}(|\mathcal{X}_{\text{GPDP}}|^3 + |\mathcal{U}|^3 \times |\mathcal{X}_{\text{GPDP}}|)$,
 $|\mathcal{X}_{\text{GPDP}}| \approx 200$ (regular grid) $\rightarrow 2.1 \times 10^8$ operations
 \rightarrow independent of time-sampling frequency!

GPDP Extensions

what happens if we have some unknowns?

- ▶ perceive noisy immediate costs ✓
- ▶ unknown transition dynamics (stochastic/deterministic) ✓
related to e.g. [system identification](#) and [robust optimal control](#)
- ▶ active learning/[optimal design](#) (efficient use of data) ✓
related to e.g. [exploration-exploitation tradeoff](#)
- ▶ online learning (dynamics, value function models) ✓
related to e.g. [adaptive control](#) and [S\(P\)LAM](#)

Wrap-up

- ▶ generalize DP by means of GPs (GPDP) to **continuous states and controls**
 - flexible, nonparametric model
 - information about model uncertainty
 - required number of support points for value function model independent of time-sampling frequency
- ▶ (nonlinear, discontinuous) policy model in entire state space
- ▶ extensions and possible links to control

<http://mlg.eng.cam.ac.uk/marc/>

mpd37@cam.ac.uk

Acknowledgements:

- ACC travel support
- German Research Foundation (DFG)

References



Christopher G. Atkeson and Stefan Schaal.

Robot Learning from Demonstration.

In D. H. Fisher Jr., editor, *Proceedings of the 14th International Conference on Machine Learning (ICML-1997)*, pages 12–20, Nashville, TN, USA, July 1997. Morgan Kaufmann.



Marc P. Deisenroth, Jan Peters, and Carl E. Rasmussen.

Approximate Dynamic Programming with Gaussian Processes.

In *Proceedings of the 2008 American Control Conference (ACC 2008)*, Seattle, WA, USA, June 2008.



Marc P. Deisenroth, Carl E. Rasmussen, and Jan Peters.

Model-Based Reinforcement Learning with Continuous States and Actions.

In *Proceedings of the 17th European Symposium on Artificial Neural Networks (ESANN 2008)*, Bruges, Belgium, April 2008.



Dirk Ormoneit and Šaunak Sen.

Kernel-Based Reinforcement Learning.

Machine Learning, 49(2–3):161–178, November 2002.



Martin Riedmiller.

Neural Fitted Q Iteration—First Experiences with a Data Efficient Neural Reinforcement Learning Method.

In *Proceedings of the 16th European Conference on Machine Learning (ECML)*, Porto, Portugal, 2005.



Carl E. Rasmussen and Christopher K. I. Williams.

Gaussian Processes for Machine Learning.

Adaptive Computation and Machine Learning. The MIT Press, Cambridge, MA, USA, 2006.



Richard S. Sutton and Andrew G. Barto.

Reinforcement Learning: An Introduction.

Adaptive Computation and Machine Learning. The MIT Press, Cambridge, MA, USA, 1998.

Algorithm: Dynamic Programming

$V_0^*(\mathcal{X}) = g_{\text{term}}(\mathcal{X})$ ▷ terminal return
for $k = 1$ to N **do** ▷ for all subproblems of length k
 for all states $\mathbf{x}_i \in \mathcal{X}$ **do**
 $Q_k(\mathbf{x}_i, \mathcal{U}) = g(\mathbf{x}_i, \mathcal{U}) + \text{E} [V_{k-1}^*(\mathbf{x}') | \mathbf{x}_i, \mathcal{U}]$ ▷ Q -values
 $V_k^*(\mathbf{x}_i) = \min_{\mathbf{u} \in \mathcal{U}} Q_k(\mathbf{x}_i, \mathbf{u})$
 $\pi_k^*(\mathbf{x}_i) = \arg \min_{\mathbf{u} \in \mathcal{U}} Q_k(\mathbf{x}_i, \mathbf{u})$
 end for
end for
return $V_N^*, \pi_N^*(\mathcal{X})$ ▷ return value function and opt. policy

Algorithm: GPDP

assume known deterministic system function $\mathbf{x}' = f(\mathbf{x}, \mathbf{u})$

$$V_0^*(\mathcal{X}) = g_{\text{term}}(\mathcal{X})$$

▷ terminal return

for $k = 1$ to N **do**

▷ start recursion

for all states $\mathbf{x}_i \in \mathcal{X}$ **do**

$$Q_k(\mathbf{x}_i, \mathcal{U}) = g(\mathbf{x}_i, \mathcal{U}) + \mathbb{E}[V_{k-1}^*(\mathbf{x}') | \mathbf{x}_i, \mathcal{U}]$$

$$\boldsymbol{\pi}_k^*(\mathbf{x}_i) = \arg \min_{\mathbf{u} \in \mathcal{U}} Q_k(\mathbf{x}_i, \mathbf{u})$$

▷ minimize

$$V_k^*(\mathbf{x}_i) = \min_{\mathbf{u} \in \mathcal{U}} Q_k(\mathbf{x}_i, \mathbf{u})$$

end for

end for

return $V_N^*, \boldsymbol{\pi}_N^*(\mathcal{X})$

Algorithm: GPDP

assume known deterministic system function $\mathbf{x}' = f(\mathbf{x}, \mathbf{u})$

$$V_0^*(\mathcal{X}) = g_{\text{term}}(\mathcal{X})$$

$$V_0^* \sim \mathcal{GP}_v(m_v, k_v)$$

▷ terminal return

▷ GP models V_0^*

for $k = 1$ to N **do**

▷ start recursion

for all states $\mathbf{x}_i \in \mathcal{X}$ **do**

$$Q_k(\mathbf{x}_i, \mathcal{U}) = g(\mathbf{x}_i, \mathcal{U}) + \mathbb{E}[V_{k-1}^*(\mathbf{x}') | \mathbf{x}_i, \mathcal{U}]$$

$$\boldsymbol{\pi}_k^*(\mathbf{x}_i) = \arg \min_{\mathbf{u} \in \mathcal{U}} Q_k(\mathbf{x}_i, \mathbf{u})$$

▷ minimize

$$V_k^*(\mathbf{x}_i) = \min_{\mathbf{u} \in \mathcal{U}} Q_k(\mathbf{x}_i, \mathbf{u})$$

end for

end for

return $V_N^*, \boldsymbol{\pi}_N^*(\mathcal{X})$

Algorithm: GPDP

assume known deterministic system function $\mathbf{x}' = f(\mathbf{x}, \mathbf{u})$

$$V_0^*(\mathcal{X}) = g_{\text{term}}(\mathcal{X})$$

$$V_0^* \sim \mathcal{GP}_v(m_v, k_v)$$

▷ terminal return

▷ GP models V_0^*

for $k = 1$ to N **do**

▷ start recursion

for all states $\mathbf{x}_i \in \mathcal{X}$ **do**

$$Q_k(\mathbf{x}_i, \mathcal{U}) = g(\mathbf{x}_i, \mathcal{U}) + m_v(\mathbf{x}')$$

$$Q_k(\mathbf{x}_i, \cdot) \sim \mathcal{GP}_q(m_q, k_q)$$

▷ GP models Q_k

$$\pi_k^*(\mathbf{x}_i) = \arg \min_{\mathbf{u} \in \mathbb{R}^{n_u}} m_q(\mathbf{u})$$

▷ minimize

$$V_k^*(\mathbf{x}_i) = \min_{\mathbf{u} \in \mathbb{R}^{n_u}} m_q(\mathbf{u})$$

end for

end for

return $V_N^*, \pi_N^*(\mathcal{X})$

Algorithm: GPDP

assume known deterministic system function $\mathbf{x}' = f(\mathbf{x}, \mathbf{u})$

$$V_0^*(\mathcal{X}) = g_{\text{term}}(\mathcal{X})$$

$$V_0^* \sim \mathcal{GP}_v(m_v, k_v)$$

▷ terminal return

▷ GP models V_0^*

for $k = 1$ to N **do**

▷ start recursion

for all states $\mathbf{x}_i \in \mathcal{X}$ **do**

$$Q_k(\mathbf{x}_i, \mathcal{U}) = g(\mathbf{x}_i, \mathcal{U}) + m_v(\mathbf{x}')$$

$$Q_k(\mathbf{x}_i, \cdot) \sim \mathcal{GP}_q(m_q, k_q)$$

▷ GP models Q_k

$$\pi_k^*(\mathbf{x}_i) = \arg \min_{\mathbf{u} \in \mathbb{R}^{n_u}} m_q(\mathbf{u})$$

▷ minimize

$$V_k^*(\mathbf{x}_i) = \min_{\mathbf{u} \in \mathbb{R}^{n_u}} m_q(\mathbf{u})$$

end for

$$V_k^* \sim \mathcal{GP}_v(m_v, k_v)$$

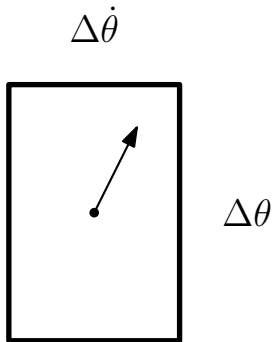
▷ GP models V_k^*

end for

return $V_N^*, \pi_N^*(\mathcal{X})$

Higher Spatial Resolution for Higher Sampling Rate

$$\ddot{\theta}\Delta_t < \Delta\dot{\theta} \quad \dot{\theta}\Delta_t < \Delta\theta$$



Computational Requirements

- GPDP: $\mathcal{O}(|\mathcal{X}|^3 + |\mathcal{U}|^3|\mathcal{X}|)$
- standard DP: $\mathcal{O}(|\mathcal{X}_{\text{DP}}|^2 \times |\mathcal{U}_{\text{DP}}|)$

Table: Computational demands and performances of GPDP and classical DP

GPDP	$ \mathcal{X} $	$\mathcal{O}_{\text{GPDP}}^{\text{comp}}$	cost	DP	$ \mathcal{X}_{\text{DP}} $	$\mathcal{O}_{\text{DP}}^{\text{comp}}$	cost
	578	2.0×10^8	9.23		6.2×10^5	9.8×10^{12}	9.02
	488	1.2×10^8	9.24		3.1×10^5	1.3×10^{12}	9.05
	400*	7.0×10^7	9.55*		2.3×10^5	6.1×10^{11}	9.05
	398	6.9×10^7	9.29		6.4×10^3	1.0×10^9	9.34
	200	1.1×10^7	9.48		1.5×10^3	6.1×10^7	11.13

*: random initialization of support points for value function/policy instead of regular grid