Extended Radial Basis Function Controller for Reinforcement Learning

Naifu Zhang  
Department of Computer Science and Technology  
Tsinghua University  
China  
nz248@cantab.net

Nicholas Capel  
Gallenco Science  
Singapore  
nicholas.rj.capel@gmail.com

Abstract: There have been attempts in model-based reinforcement learning to exploit a priori knowledge about the structure of the system. This paper introduces the extended radial basis function (RBF) controller design. In addition to traditional RBF controllers, our controller comprises of an engineered linear controller inside an operating region. We show that the learnt extended RBF controller takes on the desirable characteristics of both the linear and non-linear controller models. The extended controller is shown to retain the ability for universal function approximation of the non-linear RBF functions. At the same time, it demonstrates desirable stability criteria on par with the linear controller. Learning has been done in a probabilistic inference framework (PILCO), but could generalise to other reinforcement learning frameworks. Experimental results from the Swing-up pendulum, Cartpole, and Mountain car environments are reported.

Keywords: Probabilistic inference, controller modelling, reinforcement learning

1 Introduction

In recent years, the rise of deep learning has unlocked a new class of function approximation and representation techniques that enable easy discovery of low-dimensional features in extremely high dimensional data. This allows for systems of scales not possible before by being able to more effectively address the curse of dimensionality [1]. These techniques have naturally been applied to the field of Reinforcement Learning (RL) - a technique that allows agents to learn behaviours through interactions with an environment in order to achieve some desired outcome.

A consequence of better feature discovery is methods that scale better with high dimensional states and action spaces. Examples of these include the Atari game playing algorithm [2], Alpha Go [3], and Deepstack [4].

However, the outputs of the algorithm are opaque, and the relationship between the states and actions can not easily be understood by humans. When using these techniques for medical purposes [5] or for controlling a critical chemical plant, failure can be catastrophic, and the ability of a user to understand and trust the outputs of these systems can be key to adoption [6].

Moreover, in many reinforcement learning frameworks, the specification of a reward function is exceptionally important in achieving desired behaviours. A naive optimization with the wrong reward function can easily lead to strange behaviours. In a recent paper [7], for instance, the incentivisation of forward progress leads to robots developing strange walking gaits that, while effective, are not likely to be implemented in real life systems. The shaping of the reward function to incentivise desired behaviour is therefore more art than science.

In this paper we introduce an extended radial basis function controller for reinforcement learning. We utilize this controller with the model-based RL system of PILCO [8] to prove the concept, but the controller can in theory be used with other systems with little to no modification. This controller allows for the manual specification of a linear controller within a desired region of control, and
outside that region, smoothly switches to an arbitrary radial basis function (RBF) controller that is rich enough to model large classes of functions.

When we have a system with known local linear dynamics, we can use a powerful suite of tools from control theory to design controllers with certain highly desired properties. Examples of these might include, but are not limited to: decreased system sensitivity to input noise, increased robustness of control to poor dynamics models, decreased system sensitivity to noise in sensors, and better system recovery from impulses.

We hypothesize that this extended RBF controller will be very useful in two different kinds of systems: first, systems with general non-linear dynamics, but operating points about which stability is desired; second, systems with very well-studied local behaviour, but unknown global behaviour. Examples include combustion [9], continuous non-isothermal stirred tank reactors [10], and catalytic crackers [11].

2 Method

The formal language that we adopt is one of the Markov Decision Process (MDP) [12, 13, 14] where:

- \( x \in X \) to denote a state from a set of possible states
- \( u \in U \) to denote an action drawn from a set of possible actions
- \( P(x_{t+1}|x_t, u_t) \in P(S) \) refers to the transition dynamics, which is the probability of distribution over the next states conditioned on the previous actions and states
- \( P_0 \in P(S) \) denotes the distribution of the initial state
- \( c(x, u) \in P(\mathbb{R}) \) is a random variable representing the reward obtained when an action \( u \) is taken in state \( x \)

We introduce a policy function, \( \pi(x_t) \) such that \( u_t = \pi(x_t) \). The problem of trying to discover the ideal policy \( (\pi(x_t)) \) to achieve the best possible reward thus reduces to:

\[
\pi^*(x_t) = \arg \min_{\pi} \sum_{\forall t} \mathbb{E}[c(x_t)]
\]  

Next, we discuss the probabilistic learning framework (PILCO) in which our extended controller outlined in section 3 is learnt. We find the explicit modelling of the distribution of internal states in the feed-forward dynamics to be very helpful in providing insights into learning and convergence. Below, we focus on the most relevant and important aspects of the learning method. More details can be found in [8].

**Algorithm 1** Probabilistic Learning Algorithm

1: Set linear controller within the range of the n-ellipsoid and RBF-based controller elsewhere
2: for each training epoch do
3: Execute extended controller
4: Record collected experience
5: Learn probabilistic dynamics model \( \triangleright \) dynamics model learning
6: for each timestep do
7: Simulate system with controller \( \pi \)
8: Compute expected long-term reward \( V^\pi \) \( \triangleright \) controller evaluation
9: Optimise non-linear RBF-based controller and the axial radii of the n-ellipsoid, specified by \( \Theta^* \) \( \triangleright \) controller optimisation

The probabilistic learning is partitioned into three phases: learning the dynamics model using Gaussian processes; approximate controller evaluation; controller optimisation. This process is laid out in pseudo-code in Algorithm 1.

**Learning the Dynamics Model** The dynamics of the system could be represented by a Markov Decision Process (MDP), as in Figure 1. In our set-up, each state is measurable, observable and
fully controllable. The reward function \( c(\cdot) \) is known. The deterministic transition function \( f(\cdot) \) is however unknown, and needs to be learnt

\[
x_{t+1} = f(x_t, u_t)
\]

for state \( x_t \in \mathbb{R}^D \) and control action \( u_t \in \mathbb{R}^F \).

The probabilistic dynamics model is implemented using a Gaussian process (GP), chosen for its expressive power in capturing model uncertainty. The GP allows us to perform next-step prediction via techniques presented in [8] and [15]. We compute the mean and variance along each dimension of the predictive states distribution, obtaining a fully specified GP as shown in Equation 3.

\[
x_{t+1} \sim \mathcal{N}\left( \begin{bmatrix}
E_f[f_1(x_t, u_t)|x_t, u_t] \\
\vdots \\
E_f[f_D(x_t, u_t)|x_t, u_t]
\end{bmatrix},
\begin{bmatrix}
Var_f[f_1(x_t, u_t)|x_t, u_t] & \cdots & 0 \\
\vdots & \ddots & \vdots \\
0 & \cdots & Var_f[f_D(x_t, u_t)|x_t, u_t]
\end{bmatrix}\right)
\]

Controller Evaluation

Our key objective here is to find a deterministic controller \( \pi \) such that \( \pi(x) = u \) minimizes an expected total reward of using \( \pi \) for \( T \) timesteps.

\[
V^\pi(\Theta) = \sum_{t=0}^T E_{x_t}[c(x_t)]
\]

In order to optimise \( \pi \), we need to first compute Equation 4 using the predictive states distribution given by

\[
p(x_{t+1}) = \int \int p(x_{t+1}|x_t, u_t)p(u_t|x_t)p(x_t)dx_t du_t
\]

We thus propagate model uncertainty through the timesteps, and cascade one-step predictions to obtain long-term predictions on \( p(x_0),...,p(x_T) \). These allow us to calculate Equation 4.

Controller Optimisation

Finally, what remains is to analytically compute the gradients of \( V^\pi \) with respect to controller parameters \( \Theta \). This derivative is analytically tractable by the repeated application of chain rule, thus allowing for gradient-based non-convex optimization methods to obtain the optimal controller specified by \( \Theta^* \). The parameters of the linear controller are not part of the optimisation routine, as they are optimal by construction.

We omit further intermediary mathematical details and refer the interested reader to [15] and [16].

3 Overview of Extended Controller

In an n-ellipsoid region about the equilibrium point, we define a standard linear controller:

\[
G(x) = Wx + b
\]

where \( x \in \mathbb{R}^D \) is the state, \( W \in \mathbb{R}^{F \times D} \) is a parameter matrix of weights and \( b \in \mathbb{R}^F \) is a bias vector. In each control dimension \( d \in [1, ..., D] \), the control action is given by a linear combination of states and bias. See subsection 4.2 and Appendix B for more detailed discussions on how to obtain \( W \) and \( b \) and examples.
Outside of the n-ellipsoid representing the equilibrium region, we switch to the radial basis function (RBF) controller of the form

\[ H(x) = \sum_{i} w_i k(x, c_i) \quad (7) \]

where \( x \) is a test input, \( k(., .) \) is a general representation of the basis functions and \( w_i \)s are weights.

We choose the radial basis function because it has the universal approximation and regularization capabilities [17]. In all subsequent discussions, we use the unnormalised Gaussian basis function, i.e.

\[ k(x, c_i) = \sigma^2 \exp(-((x - c_i)^T R(x - c_i))), \]

where \( R \) is \( \text{diag}(1/l_i^2) \), which are the length scales across the individual dimensions.

In the limit that the system is at the equilibrium point or target state, the control action should be that of the linear controller. In general, the closer the system is to the target state, the more the extended controller is to approximate the linear controller. The extended controller could therefore be parameterized as a weighted average of the linear and non-linear RBF controller, with \( r(x) \) determining the dependence on either.

\[
\pi(x) = r(x) G(x) + (1 - r(x)) \sum_{i} w_i k(x, c_i) \quad (8)
\]

Naturally, \( r(x) \) is a function of the Euclidean distance \( d(x) \) of the current state \( x \) from the target state \( a \). Geometrically, \( d(x) \) denotes an n-dimensional ellipsoid whose axial radii are given by a diagonal matrix \( \Lambda \) as illustrated in Figure 2 and Equation 9c.

\[
\begin{align*}
ro(x) &= \frac{1}{(1 + d(x))^2} \quad (9a) \\
d(x) &= (x - a)^T \Lambda^{-1} (x - a) \quad (9b) \\
\Lambda &= \text{diag}(\lambda_i) \quad (9c)
\end{align*}
\]

3.1 Stability Analysis

To prove that stability is maintained, first we consider the linearization of the extended controller

\[ \pi(x) = \pi(a) + \nabla \pi(a)(x - a) + \cdots \quad (10) \]

In order to evaluate the linearization of \( \pi(x) \):

\[
\begin{align*}
\pi(x) &= r(x) G(x) + (1 - r(x)) H(x) \quad (11a) \\
\nabla \pi(x) &= \nabla r(x) G(x) + r(x) \nabla G(x) + (1 - r(x)) \nabla H(x) - \nabla r(x) H(x) \quad (11b)
\end{align*}
\]

When \( x = a \), \( r(a) = 1 \) and \( \nabla r(a) = 0 \), we can substitute \( x \) with \( a \).

\[
\begin{align*}
\pi(a) &= G(a) \quad (12a) \\
\nabla \pi(a) &= \nabla G(a) \quad (12b)
\end{align*}
\]

Therefore, as the local linearisation reduces to \( G(a) \), the extended controller is also stable about the equilibrium point.
3.2 Proof of Universal Function Approximation

We will see in this section that the extended RBF controller retains the ability for universal function approximation of the RBF controller.

First, let \( S_1(K) \) denote the function space of the RBF controller function, which has a general element \( q : \mathbb{R}^r \to \mathbb{R} \)

\[
q(x) = \sum_{i=1}^{M} w_i K \left( \frac{x - c_i}{\sigma_i} \right) \tag{13}
\]

where \( M \in \mathbb{N}, \sigma_i > 0, w_i \in \mathbb{R}, c_i \in \mathbb{R}^r \) for \( i = 1, 2, \ldots, M \), and \( K \) is an integrable bounded function such that \( K \) is continuous almost everywhere and \( \int_{\mathbb{R}^r} K(x) dx \neq 0 \) [19].

Theorem 1 from Park and Sandberg [20] (reproduced in Appendix A) shows that such a vector space \( S_1(K) \) is dense in \( L^1(\mathbb{R}^r) \).

Now, in our formulation of the controller function in Equation 8, \( H(x) \) is clearly drawn from the vector space \( S_1(K) \). With \( r(x) \) defined in Equation 9a, where \( \lambda_i \in [0, \infty), \mathbf{c} \in \mathbb{R}^r, \mathbf{w} \in \mathbb{R}^r, b \in \mathbb{R} \), and the other parameters as defined for Equation 13. In the limit as \( \lambda_i \to \infty, r(x) \to 0 \) and \( \pi(x) \to q(x) \). Hence, the extended controller retains its ability to approximate functions in the \( L^1 \) space to arbitrary accuracy by learning the right parameterization. This is a direct corollary of the proof of universal function approximation presented in [20].

**Corollary 3.0.1** \( \exists \lambda_i \in [0, \infty) \) \( \forall i \) such that \( \pi(x) \) is dense in \( L^1 \).

4 Experiments

We evaluate our proposed approach on three RL tasks: Swing-up pendulum, Cartpole, and Mountain car. Hyperparameters and implementation details are to be found in the Supplementary Material.

4.1 Environments

(a) Pendulum-v0  
(b) CartPole-v1  
(c) Modified MountainCarContinuous-v0

Figure 3: The controlled agent in its target state in each of the classical control tasks.

**Swing-up Pendulum** OpenAI’s Pendulum-v0 (Figure 3a) [21] is used to implement this classical control task. The observation space is a 2-dimensional (pole angle \( \theta \), pole velocity \( \dot{\theta} \)). The action space is a torque \( u \) applied in either the clockwise or counter-clockwise direction. The objective is to have the controller swing the pendulum up and balance it in the target state with \( (\theta, \dot{\theta}) = (0, 0) \).

**Cartpole** OpenAI’s CartPole-v1 (Figure 3b) [21] provides an environment consisting of a cart running on a track and a freely swinging pendulum attached to the cart. A non-discrete force \( u \) is applied to the cart in order to effect the state of the pendulum. The observation space is (cart position \( x \), cart velocity \( \dot{x} \), pole angle \( \theta \), pole velocity \( \dot{\theta} \)). Similar to the swing-up pendulum system, the objective is to have the controller move the pendulum up in the middle of the track and balance it in the target state with \( (x, \dot{x}, \theta, \dot{\theta}) = (0, 0, 0, 0) \).
**Mountain Car** We base the mountain car (Figure 3c) off OpenAI’s MountainCarContinuous-v0 environment [21]. The observation space is a 2-dimensional \((x, \dot{x})\). A continuous force \(u\) is applied on the car for it to travel from its starting position in the valley to not only reach but also stay at the top of the mountain, i.e. the target state is \((x, \dot{x}) = (0, 0)\).

### 4.2 Linear Controller

Each of these systems follows the state-space equation in the canonical form, where \(x\) represents the observed states, and \(y\) the output. Please refer to Appendix B for exact forms.

\[
\dot{x} = Ax + Bu \tag{14a} \\
y = Cx + Du \tag{14b}
\]

We then obtain the optimal state-feedback control gains to achieve closed-loop stable and high performance controller design. The gain matrix \(K\) is computed via the Linear Quadratic Regulator (LQR). The feedback action could thus be synthesised by the following

\[
u = -Kx \tag{15}
\]

Equation 15 is simply Equation 6 with \(W = -K\) and \(b = 0\).

We reiterate that the engineered linear controller only provides a good approximation for the region in the n-ellipsoid [22]. There have been attempts to engineer an additional linear controller for the swing-up part of the system [23]. For that, we train a linear combination of RBF controllers as discussed in section 3.

### 4.3 Results and Discussion

The experimental environments are summarised in Table 1. The size of the parameter space of each environment indicates the dimension of \(\Theta\) in Equation 4 and Algorithm 1.

| Environment       | State Space | Action Space | Trainable Parameter Space |
|-------------------|-------------|--------------|---------------------------|
| Swing-up Pendulum | \((\theta, \dot{\theta}) \in \mathbb{R}^2\) | Torque \(u \in [-2.0, 2.0]\) Nm | \(\mathbb{R}^{97}\) |
| Cartpole          | \((x, \dot{x}, \theta, \dot{\theta}) \in \mathbb{R}^4\) | Force \(u \in [-50.0, 50.0]\) N | \(\mathbb{R}^{311}\) |
| Mountain Car      | \((x, \dot{x}) \in \mathbb{R}^2\) | Force \(u \in [-30.0, 30.0]\) N | \(\mathbb{R}^{65}\) |

Figure 4 shows an example of the controller achieving convergence in the Swing-up Pendulum environment. Note that \(\theta\) has been transformed to \(\cos(\theta)\) and \(\sin(\theta)\). We see that the internal states have been stabilised at the right values, i.e. \((\cos(\theta), \sin(\theta), \dot{\theta}) = (1, 0, 0)\) (orange). The actual states (orange) are able to converge to the stable equilibrium in spite of divergence in the dynamics model because the linear controller has taken over in the operating region close to the equilibrium \((r \approx 1)\).

The n-ellipsoid with radii \(\Lambda\) has also learnt suitable values, presenting a narrower radii in the \(\cos(\theta)\) dimension for instance, agreeing with intuition as the linear controller should only activate in the region where \(\cos(\theta) \approx 1\).

We also note that our controller converges very quickly once the linear part of the extended controller is activated with \(r \approx 1\). The vanilla RBF controller, on the other hand, requires extra system interaction time to explicitly learn the dynamics about the operating region before a suitable controller can be discovered. We are therefore able to cut down interaction time needed to achieve a stable controller with the extended controller. In other words, the required experience for learning is shorter. See Figure 5.

### 4.4 Stability Analysis
Figure 4: We show the key metrics of the Swing-up Pendulum experiment when convergence is achieved. All lines in orange indicate actual observed values; All lines in blue indicate prediction by the GP model. The top row displays the evolution of the internal states across timesteps in a single epoch. The bottom row shows the linear controller ratio in the same epoch and the $\Lambda$ of the n-ellipsoid across all epochs.

In classical control theory, the stability of feedback control to parameter uncertainty is estimated by using gain and phase margins [24]. These measure the ability of the closed loop system to withstand gain and phase changes in open loop dynamics. We perform the stability margins analysis of the controlled system Swing-up Pendulum, with non-linear controllers linearised as necessary.

We make two observations from Table 2. First, that the linear controller and the extended controller share the same gain and phase margins (as proven in subsection 3.1). This serves as further evidence that control performance is preserved across the linear and extended RBF controllers within the operating region. Second, we observe that the RBF controller has poorer stability margins. This serves as empirical evidence to back up the claim made in section 1: that cost-function based approaches in end-to-end reinforcement learning can lead to less desirable equilibrium properties.

The following empirical result shows how the poorer stability margins of the RBF controller manifest in the experimental environments. For any given environment, we add various amounts of modelling errors, as a percentage drawn from a uniform distribution $U(-\text{noise}, \text{noise})$, to the inter-

Figure 5: Interaction time of our extended RBF controller vs. that of RBF controller in PILCO. Data on PILCO cited from [15] wherever available.
Table 2: Gain and Phase Margin analysis for the Swing-up pendulum system with different controllers.

| Margins   | Linear Controller | RBF Controller (PILCO) | Extended RBF Controller |
|-----------|-------------------|------------------------|-------------------------|
| Gain Margin | ∞ dB               | 0.43 dB                 | ∞ dB                    |
| Phase Margin | 78.16°             | 42.13°                  | 78.16°                  |

nal parameters of the system. We then observe the behavior of the system around the equilibrium point with different controllers. A higher percentage of stable trials corresponds to a more stable system. As in Table 2, Figure 6 demonstrates that the extended RBF controller is much more robust to system modelling uncertainties than the vanilla RBF controller.

5 Conclusion

In this paper we introduced the extended radial basis function controller. We proved two desirable properties of our extended controller - that it is stable about a desired operating point for a given system, and that it retains the ability to serve as a universal function approximator. Via a series of experiments we demonstrate the effectiveness of this new controller by showing faster system convergence with lower interaction time in the PILCO framework. We also demonstrate that it has high robustness to system modelling uncertainties as compared to the vanilla RBF controller. We thus demonstrate that our new controller is a practical method for incorporating existing model knowledge into reinforcement learning systems via manipulation of controller models about linearised operating regions.

5.1 Further Work

There are many options for future work. First, we can extend the parameterisation of the combined controller to model multiple operating regions. In the limit as we increase the number of operating regions, we hypothesize that we can incorporate not only local linear models from classical control theory, but gain scheduling [25] and other advanced techniques into a single controller for reinforcement learning. Second, we can extend our proposed controller to model-free RL frameworks. We currently adopt a model-based RL framework, but as we do not directly manipulate the model, the parameterisation of our new controller can still be used on model-free systems [26, 27]. Third, we can adopt our controller parameterisation for other model-based systems. We can, for example, incorporate the controller structure directly into a neural network based policy function and attain similar properties to those achieved in this paper.
Acknowledgments

If a paper is accepted, the final camera-ready version will (and probably should) include acknowledgments. All acknowledgments go at the end of the paper, including thanks to reviewers who gave useful comments, to colleagues who contributed to the ideas, and to funding agencies and corporate sponsors that provided financial support.

References

[1] Y. Bengio, A. Courville, and P. Vincent. Representation Learning: A Review and New Perspectives. Technical report. URL http://www.image-net.org/challenges/LSVRC/2012/results.html.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with Deep Reinforcement Learning. 12 2013. URL http://arxiv.org/abs/1312.5602.

[3] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. Van Den Driessche, T. Graepel, and D. Hassabis. Mastering the game of Go without human knowledge. Nature, 550(7676):354–359, 10 2017. ISSN 14764687. doi:10.1038/nature24270. URL https://www.nature.com/articles/nature24270.

[4] M. Moravčík, M. Schmid, N. Burch, V. Lisý, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, and M. Bowling. DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. Science, 356(6337):508–513, 5 2017. ISSN 10959203. doi:10.1126/science.aam6960. URL https://science.sciencemag.org/content/356/6337/508.abstract.

[5] A. Ferrarese, G. Pozzi, F. Borghi, A. Marano, P. Delbon, B. Amato, M. Santangelo, C. Buccelli, M. Niola, V. Martino, and E. Capasso. Malfunctions of robotic system in surgery: Role and responsibility of surgeon in legal point of view, 1 2016. ISSN 19112092. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5329842/.

[6] M. T. Ribeiro, S. Singh, and C. Guestrin. &quot;Why Should I Trust You?&quot; Explaining the Predictions of Any Classifier. 2 2016. URL http://arxiv.org/abs/1602.04938.

[7] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. Riedmiller, and D. Silver. Emergence of Locomotion Behaviours in Rich Environments. 7 2017. URL http://arxiv.org/abs/1707.02286.

[8] M. Deisenroth, C. R. o. t. t. ...., and u. 2011. PILCO: A model-based and data-efficient approach to policy search. ausy.informatik.tu-darmstadt.de. URL http://www.ausy.informatik.tu-darmstadt.de/uploads/Publications/Deisenroth_ICML_2011.pdf.

[9] A. Fuerhapter, E. Unger, W. F. Piock, and G. K. Fraidl. The new AVL CSI engine - HCCI operation on a multi cylinder gasoline engine. In SAE Technical Papers. SAE International, 3 2004. ISBN 0768013194. doi:10.4271/2004-01-0551. URL https://www.sae.org/publications/technical-papers/content/2004-01-0551/.

[10] A. Alexandridis and H. Sarimveis. Control of processes with multiple steady states using MPC and RBF neural networks, volume 29. Elsevier B.V., 2011. doi:10.1016/B978-0-444-53711-9.50140-1. URL https://linkinghub.elsevier.com/retrieve/pii/B9780444537119501401.

[11] A. Arbel, J. H. Rinard, R. Shinmar, and A. V. Sapre. Dynamics and Control of Fluidized Catalytic Crackers. 2. Multiple Steady States and Instabilities. Industrial and Engineering Chemistry Research, 34(9):3014–3026, 9 1995. ISSN 15205045. doi:10.1021/ie00048a013. URL https://pubs.acs.org/doi/pdf/10.1021/ie00048a013.
[12] R. Bellman. A Markovian decision process. Technical report. URL https://www.jstor.org/stable/24900506.

[13] M. Puterman. Markovian decision problems. 1994.

[14] M. Ghavamzadeh, S. Mannor, J. Pineau, and A. Tamar. Bayesian Reinforcement Learning: A Survey. Foundations and Trends in Machine Learning, 8(5-6):359–483, 9 2016. doi:10.1561/2200000049. URL http://arxiv.org/abs/1609.04436http://dx.doi.org/10.1561/2200000049.

[15] C. E. Rasmussen. Gaussian Processes in machine learning. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 3176:63–71, 2004. ISSN 03029743. doi:10.1007/978-3-540-28650-9[4].

[16] M. Deisenroth. Efficient reinforcement learning using Gaussian processes. 2010. URL https://books.google.com/books?hl=en&lr=&id=Vt0q--oGhZ4C&oi=fnd&pg=PR7&dq=E%3Fcient+Reinforcement+Learning+using+Gaussian+Processes&ots= syD3mSlJny&sig=zFBGgKHHnS7kaMZk1N9DCDJKPK.

[17] Y. Wu, H. Wang, B. Zhang, K. D. I. A. Mathematics, and U. 2012. Using radial basis function networks for function approximation and classification. hindawi.com. URL https://www.hindawi.com/issrn/appmath/2012/324194/citations/.

[18] Ag2gaeh. Ellipsoid 1 (with author editions), 2018. URL https://commons.wikimedia.org/wiki/File:Ellipsoid-1-tab.svg.

[19] J. Park and I. W. Sandberg. Universal Approximation Using Radial-Basis-Function Networks. Neural Computation, 3(2):246–257, 6 1991. ISSN 0899-7667. doi:10.1162/neco.1991.3.2.246. URL https://www.mitpressjournals.org/doi/abs/10.1162/neco.1991.3.2.246.

[20] J. Park and I. W. Sandberg. Approximation and Radial-Basis-Function Networks. Neural Computation, 5(2):305–316, 3 1993. ISSN 0899-7667. doi:10.1162/neco.1993.5.2.305. URL https://www.mitpressjournals.org/doi/abs/10.1162/neco.1993.5.2.305.

[21] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. 6 2016. URL http://arxiv.org/abs/1606.01540.

[22] T. Raiko and M. Tornio. Variational Bayesian learning of nonlinear hidden state-space models for model predictive control. 2009. doi:10.1016/j.neucom.2009.06.009. URL www.elsevier.com/locate/neucom.

[23] W. Zhong and H. Röck. Energy and passivity based control of the double inverted pendulum on a cart. In IEEE Conference on Control Applications - Proceedings, pages 896–901, 2001. doi:10.1109/cca.2001.973983.

[24] H. Bode. Network analysis and feedback amplifier design. 1945.

[25] D. J. Leith and W. E. Leithead. Survey of gain-scheduling analysis and design, 7 2000. ISSN 00207179.

[26] R. S. Sutton. Learning to Predict by the Methods of Temporal Differences. Technical report.

[27] C. J. C. H. Watkins and P. Dayan. Q-Learning. Technical report, 1992.
Appendix

A Universal Function Approximation Proof

Theorem 1 presented in [20] is reproduced below for reference for Corollary 3.0.1.

**Theorem A.1** Assuming that $K: \mathbb{R}^r \to \mathbb{R}$ is integrable, $S_1(K)$ is dense in $L^1(\mathbb{R}^r)$ if and only if $\int_{\mathbb{R}^r} K(x)dx \neq 0$.

B Linear Controllers in Experiments

We describe each of the linear controllers by their state-space equations about the equilibrium point. These local equations are linearly approximated wherever appropriate. All systems are frictionless. All systems follow the canonical state-space equation of Equation 14a.

\[
\begin{pmatrix}
\dot{x} \\
\ddot{x}
\end{pmatrix}
= 
\begin{pmatrix}
A & B
\end{pmatrix}
\begin{pmatrix}
x \\
\dot{x}
\end{pmatrix}
+ 
\begin{pmatrix}
u
\end{pmatrix}
\]

Swing-up pendulum

\[
\begin{pmatrix}
\dot{\theta} \\
\ddot{\theta}
\end{pmatrix}
= 
\begin{pmatrix}
0 & 1 \\
\frac{m l g}{m l^2 + I} & 0
\end{pmatrix}
\begin{pmatrix}
\theta \\
\dot{\theta}
\end{pmatrix}
+ 
\begin{pmatrix}
0 \\
-\frac{1}{m l^2 + I}
\end{pmatrix}
u
\]

Cartpole

\[
\begin{pmatrix}
\dot{x} \\
\ddot{x} \\
\dot{\theta} \\
\ddot{\theta}
\end{pmatrix}
= 
\begin{pmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & \frac{m^2 l^2 g}{I(M+m)+M m l^2} & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & \frac{m g (M+m)}{I(M+m)+M m l^2} & 0
\end{pmatrix}
\begin{pmatrix}
x \\
\dot{x} \\
\theta \\
\dot{\theta}
\end{pmatrix}
+ 
\begin{pmatrix}
0 \\
-\frac{I+M m^2 l^2}{I(M+m)+M m l^2} \\
0 \\
\frac{m l}{I(M+m)+M m l^2}
\end{pmatrix}
u
\]

Mountain car

\[
\begin{pmatrix}
\dot{x} \\
\ddot{x}
\end{pmatrix}
= 
\begin{pmatrix}
0 & 1 \\
g & 0
\end{pmatrix}
\begin{pmatrix}
x \\
\dot{x}
\end{pmatrix}
+ 
\begin{pmatrix}
0 \\
-\frac{1}{M}
\end{pmatrix}
u
\]

C Linearization of RBF functions

In order to perform margin analysis on the RBF controller, it needs to be transformed into a linear controller around the equilibrium point. The following mathematical steps outline how we linearise the RBF functions.
\( R = diag(1/l_i) \), with \( l_i \) as the length scale for each dimension.

\[
\begin{align*}
  k(x, c_1, ..., c_n) &= \sum_{\forall i} \sigma^2 \exp\left(-\frac{1}{2}(x - c_i)^T R (x - c_i)\right) \quad (17a) \\
  \nabla_x k(x, c_1, ..., c_n) &= \sum_{\forall i} \sigma^2 R (c_i - x) \exp\left(-\frac{1}{2}(x - c_i)^T R (x - c_i)\right) \quad (17b) \\
  k(x, c_1, ..., c_n) &= k(a, c_1, ..., c_n) + \nabla_x k(a, c_1, ..., c_n)^T (x - a) + \cdots \quad (17c)
\end{align*}
\]