2008

# Value Function Approximation in Reinforcement Learning using the Fourier Basis

George Konidaris
*University of Massachusetts Amherst*

Sarah Osentoski
*University of Massachusetts Amherst*

# Value Function Approximation in Reinforcement Learning using the Fourier Basis

George Konidaris Sarah Osentoski

Technical Report UM-CS-2008-19

Autonomous Learning Laboratory
Computer Science Department
University of Massachusetts Amherst
{gdk, sosentos}@cs.umass.edu

**Abstract**

We describe the Fourier Basis, a linear value function approximation scheme based on the Fourier Series. We empirically evaluate its properties, and demonstrate that it performs well compared to Radial Basis Functions and the Polynomial Basis, the two most popular fixed bases for linear value function approximation, and is competitive with learned Proto-Value Functions even though no extra experience or computation is required.

## 1 Introduction

Reinforcement learning in continuous state spaces requires function approximation. Most work in this area focuses on *linear function approximation*, where the value function is represented as a weighted linear sum of a set of features (also known as basis functions) computed from the available state variables. Linear function approximation results in a simple update rule and quadratic error surface, even though the basis functions themselves may be arbitrarily complex.

Reinforcement learning researchers have employed a wide variety of basis function schemes, most commonly Radial Basis Functions (RBFs) and CMACs (Sutton and Barto, 1998). For most problems, choosing the right basis function set is criticial for successful learning. Unfortunately, most approaches require significant design effort or problem insight, and no basis function set is both sufficiently simple and sufficiently reliable to be generally satisfactory. Recent work (Mahadevan, 2005) has focused on learning basis function sets from experience, removing the need to design a function approximator but introducing the need to gather data to create one before learning can take place.

In the applied sciences the most common continuous function approximation method is the Fourier Series. Although the Fourier Series is a simple and effective function approximator with solid theoretical underpinnings, it is almost never used for value function approximation.

In this paper we describe the Fourier Basis, a linear function approximation scheme that uses the terms of the Fourier Series as basis functions, resulting in a simple fixed basis scheme that should perform well generally. We present results that show that it performs well compared to Radial Basis Functions and the Polynomial Basis, the fixed bases most commonly used for linear value function approximation, and is competitive with learned Proto-Value Functions even though no extra experience or computation is required.

## 2 Background

A continuous-state Markov Decision Process (MDP) can be described as a tuple:

$$M = \langle S, A, P^a_{ss'}, R^a_{ss'} \rangle,$$

where $S \subset \mathbb{R}^d$ is a set of possible state vectors, $A$ is a discrete set of actions, $P^a_{ss'}$ is the transition model (modelling the probability of moving from one state to another given an action), and $R^a_{ss'}$ is the reward function.

The goal of reinforcement learning is to learn a policy $\pi$ that maps a state vector to an action so as to maximize return (discounted sum of rewards). When $P^a_{ss'}$ is known, this can be achieved by learning a *value function* $V$ that maps a given state vector to return, and selecting actions that result in the states with highest $V$. When $P^a_{ss'}$ not available, the agent must learn an *action-value function* $Q$ that maps state-action pairs to expected return. Since the theory underlying the two cases is similar in the discussions that follow we consider only the value function case.

$V$ is commonly approximated as a weighted sum of a set of given basis functions $\phi_1, ..., \phi_n$:

$$\bar{V}(\mathbf{x}) = \sum_{i=1}^{n} w_i \phi_i(\mathbf{x}). \tag{1}$$

This is termed *linear value function approximation* since learning then entails finding the set of weights $\mathbf{w} = [w_1, ..., w_n]$ corresponding to an approximate optimal value function $\bar{V}^*$, and the value function is linear in $\mathbf{w}$. Linear function approximation methods are attractive because they result in simple update rules (often using gradient descent) and possess a quadratic error surface that (except in degenerate cases) has a single minimum. In addition, they can potentially represent very complex value functions since the basis functions themselves can be arbitrarily complex.

### 2.1 The Polynomial Basis

Given a set of $n$ state variables $\mathbf{x} = [x_1, ..., x_n]$, the simplest linear scheme is to use each variable directly as a basis function along with a single constant basis function, and set $\phi_0(\mathbf{x}) = 1$ and $\phi_i(\mathbf{x}) = x_i$.

However, most interesting value functions are too complex to be represented this way. This scheme was therefore generalized to the Polynomial Basis (Lagoudakis and Parr, 2003):

$$\phi_i(\mathbf{x}) = \prod_{j=1}^{n} x_j^{m_j}, \tag{2}$$

where each $m_j$ is an integer between 0 and $k$. We describe such a basis as an order $k$ Polynomial Basis.

### 2.2 Radial Basis Functions

Another common basis scheme is Radial Basis Functions:

$$\phi_i(\mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-||c_i - \mathbf{x}||^2/2\sigma^2}, \tag{3}$$

for a given set of centers $c_i$ and variance $\sigma^2$. The centers are typically distributed evenly along each dimension, leading to $k^n$ centers for a given $k$. The parameter $\sigma^2$ can be varied but is often set to $\frac{2}{k-1}$. One advantage of RBFs is that they only generalize locally—changes in one area of the state space do not affect the entire state space. However, this limited generalization is often reflected in slow initial performance.

### 2.3 Proto-Value Functions

Recent research has focused on learning basis functions given experience. The most prominent type of learned basis functions are *Proto-Value Functions* or PVFs (Mahadevan, 2005). In their simplest form, an agent builds an adjacency matrix $A$ by experience and then computes the Laplacian $L$ of $A$:

$$L = (D - A), \tag{4}$$

where $D$ is a diagonal matrix with $D(i, i)$ being the out-degree of state $i$. The eigenvectors of $L$ form a set of bases that respect the topology of the state space (as reflected in $A$), and can be used as a set of orthogonal bases for a discrete domain.

Mahadevan *et al.* (2006) extended PVFs to continuous domains, where a local distance metric is required to construct the graph and an out-of-sample method (which may be expensive to compute) is required for obtaining the values of each basis function at states not represented in $A$. Although the given results are promising, PVFs in continuous spaces require not only a large set of samples to produce the adjacency matrix and an expensive eigenvector decomposition to build the basis functions, but also require the programmer to make several potentially difficult design decisions.

## 3 The Fourier Basis

In this section we describe the Fourier Series for one and multiple variables, and use it to define the univariate and multivariate Fourier Bases.

### 3.1 The Univariate Fourier Series

The $n$th degree Fourier expansion of function $f(x)$ with period $T$ is given by:

$$\bar{f}(x) = \frac{a_0}{2} + \sum_{k=1}^{n} \left[ a_k \cos(k\frac{2\pi}{T}x) + b_k \sin(k\frac{2\pi}{T}x) \right], \tag{5}$$

where

$$a_k = \frac{2}{T} \int_0^T f(x) \cos(k\frac{2\pi}{T}x)\, dx \tag{6}$$

$$b_k = \frac{2}{T} \int_0^T f(x) \sin(k\frac{2\pi}{T}x)\, dx. \tag{7}$$

For the remainder of this paper we assume for simplicity that $T = 2$, with the variables of the function we wish to approximate scaled appropriately.

In the reinforcement learning setting $f$ is unknown and so we cannot compute $a_0, ..., a_n$ and $b_1, ..., b_n$, but we can consider them coefficients to be learned in a linear function approximation scheme, setting:

$$\phi_i(x) = \begin{cases} 1 & i = 0 \\ \cos(\frac{i+1}{2}\pi x) & i > 0, i \text{ odd} \\ \sin(\frac{i}{2}\pi x) & i > 0, i \text{ even.} \end{cases}$$

Thus a full $n$th order Fourier approximation to a one-dimensional value function results in a linear function approximator with $2n + 1$ terms. However, as we shall see below, we can usually use only $n + 1$ terms.

## 3.2 Even, Odd and Non-periodic Functions

If $f$ is known to be even (that is, $f(x) = f(-x)$, so that $f$ is symmetric about the $y$-axis), then $\forall i > 0, a_i = 0$ and we do not need to include the $\sin$ terms. This results in a value function that is *guaranteed* to be even, and reduces the number of terms required for an $n$th order Fourier approximation to $n + 1$. Similarly, if $f$ is known to be odd (that is, $f(x) = -f(-x)$, so that $f$ is symmetric with respect to the origin) then $\forall i > 0, b_i = 0$ and we can omit the $\cos$ terms.

However, in general value functions are not even, odd, or periodic (or at least not known to be in advance). In such cases, we can define our approximation over $[-1, 1]$ but only project the relevant variable to $[0, 1]$. This results in a function that is periodic on $[-1, 1]$ but unconstrained on $(0, 1]$. Moreover, we are now free to choose whether or not the function is even or odd over $[-1, 1]$ so that we can drop half of the terms in the approximation.

In general, we expect that it will be better to use the "half-odd" approximation and drop the $\sin$ terms because it results in only a slight discontinuity at the origin. Thus, we define the univariate $k$th order Fourier Basis as:

$$\phi_i(x) = \cos(i\pi x), \tag{8}$$

for $i = 0, ..., k$.

## 3.3 The Multivariate Fourier Series

The Fourier expansion of the multivariate function $F(\mathbf{x})$ with period $T$ in $m$ dimensions is:

$$\bar{F}(\mathbf{x}) = \sum_{\mathbf{c}} \left[ a_{\mathbf{c}} \cos(\frac{2\pi}{T}\mathbf{c}.\mathbf{x}) + b_{\mathbf{c}} \sin(\frac{2\pi}{T}\mathbf{c}.\mathbf{x}) \right], \tag{9}$$

where $\mathbf{c} = [c_1, ..., c_m]$, $c_i \in [0, ..., n]$, $0 \leq i \leq m$. This results in $2(n + 1)^m$ basis functions for an $n$th order full Fourier approximation to a value function in $m$ dimensions, which can be reduced to $(n + 1)^m$ if we drop either the $\sin$ or $\cos$ terms for each variable as described above.

We thus define the $k$th order Fourier Basis for $m$ variables:

$$\phi_i(\mathbf{x}) = \cos(\pi \mathbf{c}^i \cdot \mathbf{x}), \tag{10}$$

where $\mathbf{c}^i = [c_1, ..., c_m]$, $c_j \in [0, ..., k]$, $0 \leq j \leq m$. Each basis function thus has a coefficient vector $\mathbf{c}$ that attaches an integer coefficient (less than or equal to $k$) to each variable in $\mathbf{x}$; the basis set is obtained by systematically varying the variables in $\mathbf{c}$. This basis has the benefit of being easy to compute accurately even for high degrees, since $\cos$ is bounded in $[-1, 1]$, and its arguments are formed by multiplication and summation rather than exponentiation.

Although the Fourier Basis seems like a natural choice for value function approximation, we know of very few instances (e.g., Kolter and Ng (2007)) of its use in reinforcement learning, and no empirical study of its properties.

## 3.4 Variable Coupling

When applying reinforcement learning to continuous domains it is common to use an approximation scheme that assumes that each variable contributes independently to the value function. For example, an order 1 Polynomial Basis function contains $\phi$ terms for 1, $x_1$, $x_2$ and $x_1x_2$; assuming that $x_1$ and $x_2$ contribute independently means that we can drop the $x_1x_2$ term. We call such a basis *uncoupled.*

For higher order function approximators, uncoupling removes the curse of dimensionality: for an order $k$ Polynomial Basis with 4 variables, the full basis has $(k+1)^4$ terms, whereas the uncoupled basis has only $4(k+1)$ terms. Although this can lead to bad approximations (because in many cases the variables do not contribute independently) in many continuous domains it does not significantly degrade performance. However, a more sophisticated approach would involve the use of prior knowledge about which variables are likely to interact to obtain a smaller but still accurate function approximator.

The Fourier Basis facilitates this through constraints of $\mathbf{c}$, the coefficient vector used in equation 10. For example, we can obtain an uncoupled basis by requiring that only one element of $\mathbf{c}$ is ever non-zero. Alternatively, if we know that the interaction of variables $x_i$ and $x_j$ is important for learning then we can constrain $\mathbf{c}$ so that only $c_i$ and $c_j$ can be non-zero simultaneously.

We expect that most domains in which a decoupled basis performs badly are actually *weakly coupled*, in that we can use very low-order terms for the interaction between variables and high-order order terms for each variable independently. The Fourier Basis lends itself naturally to implementing such a scheme.

## 4 Empirical Evaluation

### 4.1 The Swing-Up Acrobot

The acrobot is an underactuated two-link robot, show in Figure 1, where the first link is suspended from a point and the second is able to exert torque. The goal of the swing-up acrobot task is to swing the tip of the acrobot's second joint one unit above the level of its suspension, much like a gymnast hanging from a pole and swinging above it using their waist. Since the acrobot is underactuated it cannot do this directly; instead it must swing back and forth to gain momentum.

The resulting task has four continuous variables (an angle and a velocity for each joint) and three possible actions (exerting a negative, positive or zero unit of torque on the middle joint). Sutton and Barto (1998) contains a more detailed description.

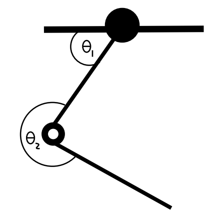


Figure 1: The Acrobot.

We employed Sarsa($\lambda$) ($\gamma = 1.0$, $\lambda = 0.9$, $\epsilon = 0$) with Fourier Bases of orders 3 (resulting in $256$ basis functions), $5$ (resulting in $1296$ basis functions) and 7 (resulting in $4096$ basis functions) and RBF, Polynomial and PVF bases of equivalent sizes to empirically compare their performance on the Acrobot task. (We did not run PVFs with $4096$ basis functions because the nearest-neighbour calculations for a graph that size proved too expensive) We systematically varied $\alpha$ (the gradient descent term) to obtain the best performance for each basis function type and order combination. The resulting $\alpha$ values are shown in Table 1.

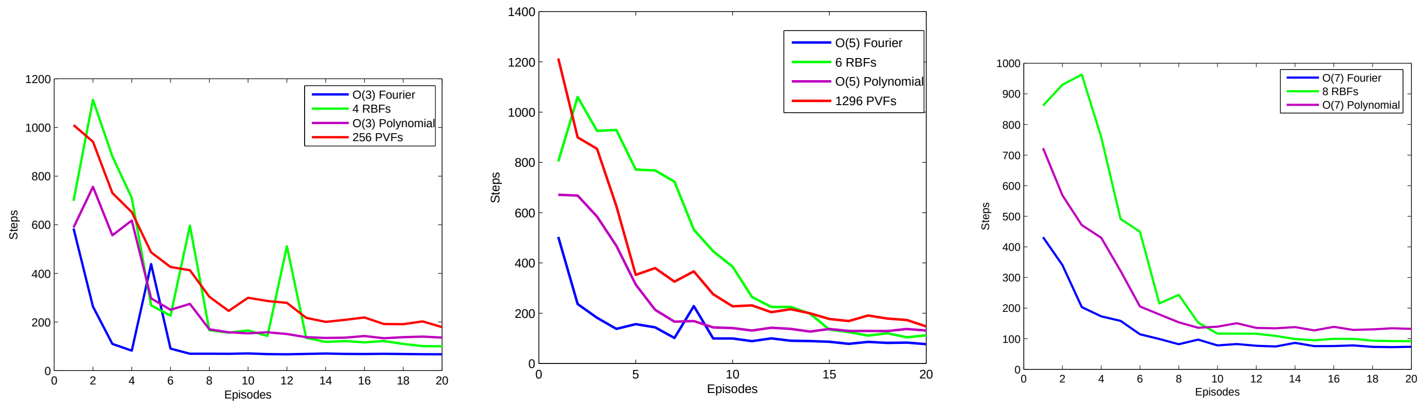
The PVFs were built using the Normalized Laplacian: $L = D^{-1/2}(D - W)D^{-1/2}$, scaling the resulting eigenvectors so that they were in the range $[-1, 1]$. We also rescaled by Acrobot state variables by $[1, 1, 0.05, 0.03]$ for local distance calculations. We found that a random walk did not suffice to collect samples that adequately cover the state space, and thus biased the sample collection to only keep examples where the random walk actually reached the target within $800$ steps. We kept $50$ of these episodes, leading to a sample size of approximately $3200$ samples. We subsampled these initial samples to graphs of $1200$ or $1400$ points, created using nearest $30$ neighbors. We did not optimize these settings as well as might have been possible, and thus the PVF averages are not as good as they could have been. However, our intention was to consider the performance of each method as generally used, and a full exploration of the parameters of PVF methods (or better construction methods, e.g. Johns *et al.* (2007); Mahadevan and Maggioni (2007)) was beyond the scope of this work.

We ran $20$ episodes averaged over $100$ runs for each type of learner. The results, shown in Figure 2, demonstrate that the Fourier Basis learners outperformed all other types of learning agents for all sizes of function approximators.

| **Basis** | **O(3)** | **O(5)** | **O(7)** |
|---|---|---|---|
| Fourier | 0.001 | 0.0005 | 0.0001 |
| RBF | 0.0075 | 0.001 | 0.00075 |
| Poly | 0.0075 | 0.0075 | 0.01 |
| PVF | 0.0075 | 0.0075 | — |

Table 1: $\alpha$ values used for the Swing-Up Acrobot.

In particular, the Fourier Basis performs better initially than the Polynomial Basis (which generalizes broadly) and converges to a better policy than the RBF Basis (which generalizes locally). It also performs slightly better than PVFs, even though the Fourier Basis is a fixed, rather than learned, basis.



(a) (b) (c)

Figure 2: Learning curves for agents using (a) order 3 (b) order 5 and (c) order 7 Fourier Bases, and RBFs and PVFs with corresponding number of basis functions.

However, the value function for Acrobot does not contain any discontinuities, and is therefore not the type of problem that PVFs are designed to solve. In the next section, we consider a continuous domain with an obstacle that induces a discontinuity in the value function.

## 4.2 The Discontinuous Room

In the Discontinuous Room, show in Figure 3, an agent in a room must reach a target; however the direct path to the target is blocked by a wall, which the agent must go around. The room measures 10x6 units, and the agent has four actions which move it 0.5 units in each direction. This domain is intended as a simple continuous domain with a large discontinuity, to empirically determine how the Fourier Basis handles such a discontinuity compared to Proto Value Functions, which were specifically designed to handle discontinuities by modelling the connectivity of the state space.

In order to make a reliable comparison, agents using PVFs were given a perfect adjacency matrix containing every legal state and transition, thus avoiding sampling issues and removing the need for an out-of-sample extension method.

Figure 4 shows learning curves for agents in the Discontinuous Room learning using Sarsa($\lambda$) ($\gamma = 0.9$, $\lambda = 0.9$, $\epsilon = 0$) using Fourier Bases of order 5 ($\alpha = 0.025$) and order 7 ($\alpha = 0.025$) and agents using 36 ($\alpha = 0.025$) and 64 ($\alpha = 0.025$) PVFs. The agents using the Fourier Basis do not initially perform as well as the agents using PVFs, probably because of the discontinuity, but this effect is transient and the Fourier Basis agents eventually perform better.

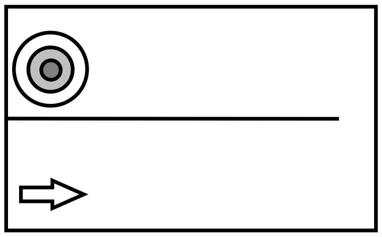


Figure 3: The Discontinuous Room, where an agent must move from the lower left corner to the upper left corner by skirting around a wall through the middle of the room.

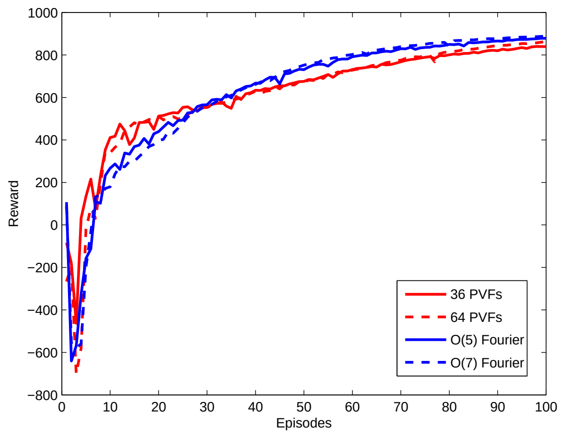


Figure 4: Learning curves for the Discontinuous Room, for agents using $O(5)$ and $O(7)$ Fourier Bases and agents using $36$ and $64$ PVFs.

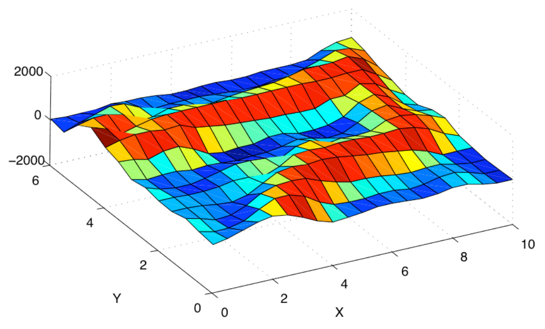
Figure 5 shows example value functions for for an agent using an order $7$ Fourier Basis and a corresponding agent using $64$ PVFs. The PVF value function is clearly a better approximation to an optimal policy value function, and in particular shows very precise modelling of the discontinuity around the wall. In contrast, the Fourier Basis value function is noisier and does not model the discontinuity as cleanly.

However, the results in Figure 4 suggest that this does not significantly impact the quality of the resulting policy, and it appears that the Fourier Basis agent has learned to avoid the discontinuity. This suggests that, at least for continuous problems with a small number of discontinuities, the Fourier Basis is sufficient, and that the extra complexity (in terms of samples and computation) of PVFs only becomes really necessary in more complex problems.

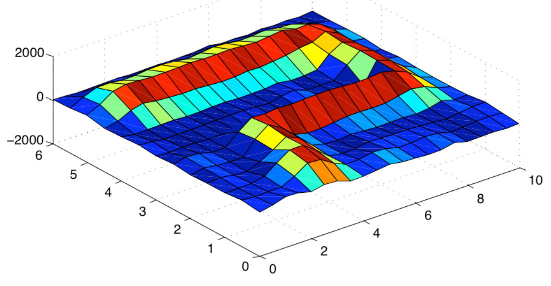
## 4.3 Mountain Car

The Mountain Car (Sutton and Barto, 1998), as shown in Figure 6 is an underpowered car that must drive up a steep cliff in a valley; to do so, it must first accelerate up the back of the valley, and then use the resulting momentum to drive up the other side. This induces a steep discontinuity in the value function which makes learning difficult, which we use to examine the properties of the Fourier Basis in a domain very difficult for bases with global support.

We employed Sarsa($\lambda$) ($\gamma = 1.0$, $\lambda = 0.9$, $\epsilon = 0$) with Fourier Bases of orders 3 and 5, and RBFs and PVFs of equivalent sizes (we were unable to learn with the Polynomial Basis). Here we found it best to scale the $\alpha$ values for the Fourier Basis by $\frac{1}{1+m}$, where $m$ was the maximum degree of the basis function. This allocated lower learning



(a)



(b)

Figure 5: Value functions for the Discontinuous Room from agents using (a) an $O(7)$ Fourier Basis and (b) 64 PVFs.

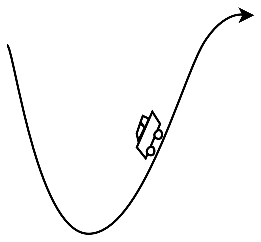


Figure 6: The Mountain Car.

rates to higher frequency basis functions. The $\alpha$ values used for learning are shown in Table 2.

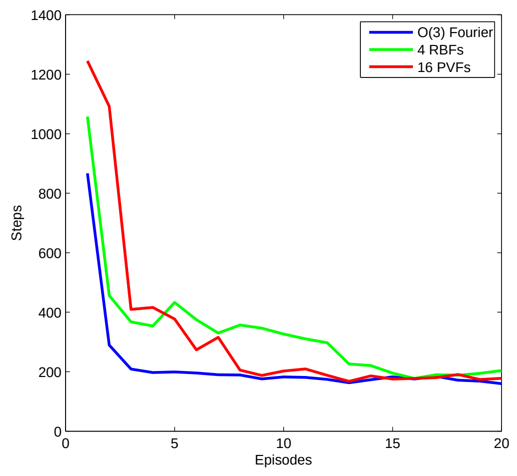
| **Basis** | **O(3)** | **O(5)** |
|---|---|---|
| Fourier | 0.025 | 0.025 |
| RBF | 0.025 | 0.025 |
| PVF | 0.01 | 0.025 |

Table 2: $\alpha$ values used for Mountain Car.

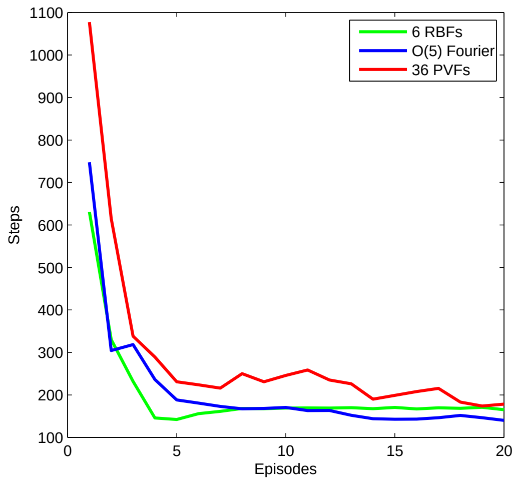
The results (shown in Figure 7) indicate that for low orders, the Fourer Basis outperforms RBFs and PVFs. For higher orders, we see a repetition of the learning curve in Figure 4, where the Fourier Basis initially performs worse (because it does not model the discontinuity in the value function well) but converges to a better solution. Thus, although the Fourier Basis is not always the best choice for domains with discontinuities, it works reliably.

## 5 Discussion

Our results show that the Fourier Basis provides a simple and reliable basis for value function approximation. We expect that for many problems, the Fourier Basis will be sufficient to learn a good value function without any extra work. If necessary, the ability to include prior knowledge about symmetry or variable interaction provides a simple and useful tool to induce structure in the value function.



(a)



(b)

Figure 7: Learning curves for agents using (a) order 3 (b) order 5 Fourier Bases, and RBFs and PVFs with corresponding number of basis functions.

The one area in which we find that the Fourier Basis has difficulty representing is flat areas in value functions; moreover, for high order Fourier Basis approximations sudden discontinuities may induces "ringing" around the point of discontinuity, known as *Gibbs phenomenon* (Gibbs, 1898). This may result in policy difficulties near the discontinuity.

Another challenge posed by the Fourier Basis (and all fixed basis function approximators) is how to select basis functions when a full order approximation of reasonable size is too large to be useful. PVFs, by contrast, scale with the size of the sample graph (and thus could potentially take advantage of the underlying complexity of the manifold). It may be possible to do something similar by sampling and then using the properties of the resulting data to select a good set of Fourier Basis functions for learning.

Another potential difficulty we have encountered is that the use of a single $\alpha$ term in gradient-descent based algorithms (such as Sarsa($\lambda$)) creates difficulties for very high-order Fourier Basis function approximators. This is because a single $\alpha$ must be found that is able to handle basis functions of very different frequencies. This difficulty will affect PVFs and the Polynomial Basis, but not Radial Basis functions because each RBF is the same size. Least-squares based methods are not affected by this problem.

Nevertheless, our experiences with the Fourier Basis show that this simple and easy to use basis set reliably performs well on a range of different problems. As such, although it may not perform well for domains with several significant discontinuities, its simplicity and reliability suggest that the Fourier Basis should be the first function approximator used when reinforcement is applied to a problem.

## 6 Summary

We have described the Fourier Basis, a linear function approximation scheme using the terms of a Fourier series as features. Our experimental results show that this simple and easy to use basis performs well compared to two popular fixed bases and a learned set of basis functions in two continuous reinforcement learning benchmarks, and is competitive with a popular learned basis even though no extra experience or computation is required.

## References

Gibbs, J. (1898). Fourier series. *Nature*, **59**(200).

Johns, J., Mahadevan, S., and Wang, C. (2007). Compact spectral bases for value function approximation using kronecker factorization. In *Proceedings of the Twenty-second National Conference on Artificial Intelligence (AAAI-07)*.

Kolter, J. and Ng, A. (2007). Learning omnidirectional path following using dimensionality reduction. In *Proceedings of Robotics: Science and Systems*.

Lagoudakis, M. and Parr, R. (2003). Least-squares policy iteration. *Journal of Machine Learning Research*, **4**, 1107–1149.

Mahadevan, S. (2005). Proto-value functions: Developmental reinforcement learning. In *Proceedings of the Twenty Second International Conference on Machine Learning*.

Mahadevan, S. and Maggioni, M. (2007). Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *Journal of Machine Learning Research*, **8**, 2169–2231.

Mahadevan, S., Maggioni, M., Ferguson, K., and Osentoski, S. (2006). Learning representation and control in continuous markov decision processes. In *Proceedings of the Twenty First National Conference on Artificial Intelligence*.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.