Adaptive Value Function Approximation in Reinforcement Learning using Wavelets

Michael Mitchley
Supervised by Prof. George Konidaris and Prof. Ebrahim Momoniat

A thesis presented for the degree of
Doctor of Philosophy

School of Computational and Applied Mathematics
University of the Witwatersrand, Johannesburg
South Africa
July, 2015

Abstract

Reinforcement learning agents solve tasks by finding policies that maximise their reward over time. The policy can be found from the *value function*, which represents the value of each state-action pair. In continuous state spaces, the value function must be approximated. Often, this is done using a fixed linear combination of functions across all dimensions.

We introduce and demonstrate the wavelet basis for reinforcement learning, a basis function scheme competitive against state of the art fixed bases. We extend two online adaptive tiling schemes to wavelet functions and show their performance improvement across standard domains. Finally we introduce the Multiscale Adaptive Wavelet Basis (MAWB), a wavelet-based adaptive basis scheme which is dimensionally scalable and insensitive to the initial level of detail. This scheme adaptively grows the basis function set by combining across dimensions, or splitting within a dimension those candidate functions which have a high estimated projection onto the Bellman error. A number of novel measures are used to find this estimate.

Dedication

To my shining star, Miriella, whose boundless love, patience, help and support has made this thesis possible.

Declaration

I, Michael Mitchley, declare that this thesis titled 'Adaptive Value Function Approximation in Reinforcement Learning using Wavelets' and the work presented in it are my own. I confirm that:

- This work was done wholly while in candidature for a research degree at this University.
- This work has not been submitted to any other institution or for any other degree.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed: [illegible]

Date: 30/07/2015

Acknowledgements

I would like to thank my supervisors George Konidaris and Ebrahim Momoniat for the help and advice offered over the past three years. In particular, I would like to thank George for the weekly meetings, for his hospitality during my visit, and for his endless enthusiasm for this project. In the long, dark teatime of the soul that happens so often during experimentation, George was always there to offer advice and support (and to tell me to improve my results).

Thank you to Philip Thomas and Bruno da Silva (and their partners) for hosting me during the February 2014 Amherst trip, and for sharing their research and favourite watering hole. I look forward to future collaborations and visits, with the next round being on me.

Thank you to Byron, Charis, and Rhameez for enduring countless small questions with good grace and making the office a fun place to work.

Contents

List of Figures

List of Tables

Notation

$\mathcal{S}$	The state space, or set of all possible states
s	An arbitrary state
π	An arbitrary policy, mapping each state to an action
π^*	The optimal policy, maximising reward
s'	The successor state of s, after executing action a according to policy π
$\mathcal{A}$	The action space, or set of all possible actions
a	An arbitrary action
a'	The action chosen in state s' using policy π
$\mathcal{T}$	A transition probability density function
γ	The discount factor of the MDP
α	The learning rate of the agent
ϵ	The exploration rate of the policy
Λ	The eligibility trace
$\underline{w}$	The parameter or weight vector of a basis function series
D	The number of dimensions in $\mathcal{S}$ for a continuous state MDP
d	A specific state dimension in a continuous state MDP
n	The order of the function approximation
ϕ	A father wavelet function in one dimension
ψ	A mother wavelet function in one dimension
Φ	A father wavelet tile in multiple dimensions
Ψ^G	A wavelet tile in all dimensions
τ	A tolerance
ρ	The per-function relevance
A	The per-function absolute error
O	The per-function observed error
T	The number of samples used in an estimate
σ	Standard deviation

Chapter 1

Introduction

When one learns a new task, one discovers the dependencies between task elements individually, rather than assuming full dependence between all task elements. Consider, for example, learning to drive: a new driver must be instructed on the interplay between the gear lever and the clutch as, in the absense of prior experience, the driver may not know of their connection. Appropriate levels of detail, too, must be discovered for the task. In our car scenario, one must discover how much to turn the steering wheel: a fraction of a degree is too fine a detail level, while 'clockwise or anticlockwise' would be too coarse. Likewise, it is sufficient to gauge the rough level of fuel in the tank (to, perhaps, the nearest quarter), rather than knowing the exact quantity of fuel remaining, or differentiating only between empty and not empty. Our first-time driver, however, might find it advantageous to begin the lesson with only this coarse level of detail, and add to it as the task is mastered.

When an intelligent computational agent learns a task, the detail level required to represent the knowledge needed to solve that task is typically set manually prior to learning, and full feature dependence is assumed between the state dimensions (where each dimension corresponds to a different measurable quantity in the environment or state). This leads to agents which do not scale with dimensionality: the number of features required to represent the same level of detail across all state dimensions grows exponentially in the number of dimensions, a problem often referred to as the curse of dimensionality. Furthermore, the initial detail chosen may be too coarse (in which case the agent may not learn to perform the task well, or could even fail to accomplish the task at all), or too fine (in which case the agent will take a long time to learn, as the knowledge gained in each attempt is highly localised). Scaling with dimension will require a representational framework in which the agent may assume independence between task features, until such dependencies are discovered to be important. Detail-based adaptivity would ideally balance generality (and thus fast learning) against the eventual representation of fine-scale detail as optimal behaviour is achieved.

Within reinforcement learning, adaptive value function approximation techniques have attempted to address the curse of dimensionality or adaptive representation separately. To date, no online linear value function approximation method has addressed both problems. Furthermore, prior adaptive techniques rely on simple binary functions, and thus lack representational ability.

In this thesis, we propose two algorithms for real-valued functions; Incremental Feature Dependency Discovery and Adaptive Wavelet Refinement, which address the problems of feature dependence discovery and detail level discovery respectively, using a number of novel measures of feature error and relevance. We then combine these into a hybrid algorithm which is able to discover appropriate dependencies and detail levels through online sampling, called the Multiscale Adaptive Wavelet Basis.

The remainder of this thesis is organised as follows. In Chapter 2, we present a literature review of topics related to value function approximation within reinforcement learning. Value functions give the 'value' of being in a particular state (or state-action pair). When the state space of the problem is continuous, the value of each state cannot be represented directly, and so an approximation of the value function is required. In Chapter 3, we discuss function approximation using wavelets, a class of refineable function which generalises Fourier analysis. Since much of the literature on wavelets discusses their use from a function analysis perspective, we explain clearly their use in the context of function synthesis, with specific focus on their use in value function approximation in reinforcement learning. We demonstrate the competitiveness of a number of fixed wavelet bases against the state of the art. These chapters may be skipped by readers familiar with either value function approximation or wavelet synthesis theory respectively.

In Chapter 4, we examine how the usefulness of a function can be measured in an online setting, and present a number of novel measures of function error and relevance. We argue for the use of these relevance measures, supported by theoretical and empirical results. In Chapter 5, the novel measures of the previous chapter are used to extend the adaptive tile coding algorithm (ATC) [Whiteson *et al.* 2007], using the wavelet basis discussed in Chapter 3. We show that the resulting algorithm is convergent, and that wavelets are both necessary and sufficient for certain classes of detail-based value function adaptation. We show also that the novel algorithm performs better than a fixed basis.

Chapter 6 extends the binary feature-based incremental feature dependency discovery algorithm (IFDD) [Geramifard *et al.* 2011] to arbitrary function types through the use of our novel relevance measures. We show convergence results and demonstrate the performance of our new algorithm against two variants of IFDD using both tile coding and wavelets. Chapter 7 showcases the main result of this thesis, the novel Multiscale Adaptive Wavelet Basis (MAWB) which combines the extensions of the previous two chapters into a basis function scheme that adapts in both feature dependency and detail level. This allows one to apply value function approximation to arbitrarily high dimensional domains at a very low initial detail level. MAWB retains the convergence properties and performance of the two algorithm extensions. We demonstrate the scalability of MAWB using a domain with 100 continuous state variables. Finally, in Chapter 8, we summarise the results and contributions, and discuss future directions that may be taken.

Chapter 2

Background and Related Work

2.1 Introduction

Before introducing the novel work of this thesis, we first provide a brief introduction to reinforcement learning, and examine the state of the art of adaptive value function approximation. This chapter is intended for readers familiar with machine learning, but readers already familiar with reinforcement learning may wish to skip to the next chapter.

2.2 Reinforcement Learning

Reinforcement learning is a machine learning framework in which an agent acts in an environment and receives feedback in the form of rewards based on the outcomes of those actions. The agent may be learning to complete a task, but that task is not explicitly programmed into the agent. Instead, the agent learns to maximise the reward it obtains—desirable behaviour is rewarded and undesirable behaviour is penalised.

Reinforcement learning tasks are commonly modelled as Markov decision processes (MDPs), although non-Markov frameworks exist as well. An MDP is described by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, R, \gamma)$ where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions the agent has available to it, the transition probability $\mathcal{T}(s, a, s') \mapsto [0, 1]$ is the transition function, or probability of transitioning to state s' after executing action a in state s, and $R(s, a, s') \mapsto \mathbb{R}$ is the reward given by the system upon transition from state s to s' through the execution of action a. The discount factor $\gamma \in (0, 1)$ informs the agent of how much future rewards should be discounted, in favour of present rewards. The goal of the agent is to maximise the total reward it obtains by finding and following an optimal policy $\pi(s,a) \mapsto [0, 1]$ giving the probability of executing action a in state s, subject to the discount γ.

If reinforcement learning were applied to the game of checkers, for example, the state would list the position of each piece on the board, and the available actions at each state would be the legal moves that could be made. If a positive reward were associated with winning, and a negative reward with losing, an agent could, given sufficient time and an appropriate framework, learn how to play checkers. Checkers has a discrete state space (although it is very large), because there are a finite number of possible positions. The game of checkers has the Markov property, whereby an observation of the current state is sufficient to make a move, as the conditional probability of future states is dependent only on the current state, and is not conditional on the state history. One would not need to know the history of the game (although for human players this history may inform us of intent).

The state space may be discrete (with a finite number of states) or continuous (with an infinite number of possible states). Likewise, the action space may be either discrete, or continuous. The state space may be partially observable, wherein the agent's information about the current state is incomplete. This violates the Markov property, as one would require prior states in addition to the current state to determine the next move.

Transitions in the MDP may be deterministic (in which case, $\mathcal{T}$ is a manifold with $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$), or stochastic, such that $\mathcal{T}$ is a set of probabilities over the state space in the discrete state case, or a probability density function in the continuous state case.

The rewards are either deterministic or stochastic, and are an intrinsic property of the MDP. The sum of discounted future rewards of the agent is given by

$$\mathcal{R} = \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}), \tag{2.1}$$

where s_i is the ith state visited by the agent. This is called the *return*, and is the quantity the agent will attempt to maximise. If the task is *episodic* (guaranteed to come to an end in a finite number of time steps), then $0 < \gamma \leq 1$, otherwise $0 < \gamma < 1$. Tasks may be *stationary*, where the reward and transition probabilities do not change, or non-stationary, where they do. If the rewards or transitions change, the underlying task we wish the agent to solve changes, and so the agent should adapt to the new task.

The transition function defining the state dynamics is typically not known to the agent. Problem frameworks wherein the transition function $\mathcal{T}$ or reward function $\mathcal{R}$ are known (or can be estimated) are known as *model-based* approaches, while those wherein these functions are not known or estimated are called *model free*. As the task to be solved is not explicitly programmed into the agent, reinforcement learning is well-suited to highly complex tasks where the optimal behaviour is unknown in advance. A common form of reward in episodic tasks, for example, is simply to penalise the agent for each time step taken until the goal state is reached. This induces the agent to complete the task as quickly as possible, in order to minimise the total penalty (which is equivalent to maximising its total reward). The complexity of the required task is no longer a factor in creating the agent, as the agent does not require foreknowledge of the required behaviour or the state dynamics (although clearly this complexity may still be an issue which affects how long the agent takes to complete the task).

The central problem of reinforcement learning is finding the policy quickly and efficiently without requiring prior knowledge of the task. This should be done without an exhaustive search through the state space, and without exorbitant memory requirements. One approach to finding this policy is through a *value function*.

The agent seeks to maximise its return (the discounted sum of future rewards). Since the return is dependent on both the current state and the policy being followed, it is useful to define the *value* of each state $V_\pi(s)$ under the policy π as

$$V_\pi(s) = \mathbb{E}\left\{ \sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t), s_{t+1}) \,|\, s_0 = s \right\}, \tag{2.2}$$

which is the expected return of following π from the current state. Note that this is similar to equation 2.1, with a prescribed initial state $s_0 = s$ and actions chosen according to policy π. This is the *state value function*. Deriving a policy from the state value function requires $\mathcal{T}$. In model-free approaches, the *state-action value function* $Q_\pi(s, a)$ is used, defined as

$$Q_\pi(s,a) = \mathbb{E}\left\{ R(s, a, s_0) + \sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t), s_{t+1}) \right\}, \tag{2.3}$$

which is the expected return of executing action a in state s, and thereafter following policy π.

It is relatively easy to derive a policy once the value $Q(s, a)$ of a state-action pair has been established. The greedy policy $\pi(s) = \arg\max_a Q(s, a)$ always selects the action with the highest value. If one wishes to explore the state space through occasional random action selection, ϵ-greedy policies select a random action with probability ϵ but otherwise select the action greedily. These have the disadvantage that an exploratory move could result in the selection of an action which is known to be very bad. Softmax [Sutton and Barto 1998] weights the action selection by the value using (commonly) a Gibbs distribution, choosing action a with probability

$$\frac{\exp \frac{Q(s,a)}{\tau}}{\sum_i \exp \frac{Q(s,i)}{\tau}},$$

where τ is a parameter functioning much like a *temperature* in simulated annealing. As τ tends to infinity, softmax turns into uniform random selection, each action being equally likely. A very low τ tending to zero reduces softmax to the greedy policy.

In both the ϵ-greedy policy and softmax, it may be beneficial to vary the parameter (ϵ and τ, respectively) with time, starting high to encourage exploration and reducing it so as to tend towards the greedy policy. In continuous action problems, finding $\arg\max_a Q(s, a)$ involves a one-dimensional search through a, although this will depend on how $Q(s, a)$ is represented.

2.3 Value Function Approximation

When the state space $\mathcal{S}$ is large or continuous, it is impossible to compute and store a value for each state. In this case, the value function must be approximated. One such approach is linear value function approximation, which represents V^π as a weighted sum of a collection of n basis functions Φ:

$$V^\pi(s) \approx \underline{w} \cdot \Phi(s) = \sum_{i=1}^{n} w_i \phi_i(s),$$

which map states to values. This approximation is linear in the components of the parameter (or weight) vector, $\underline{w}$, which results in simple update rules for the weights and a quadratic error surface, as the local minimum of the difference between $V^\pi(s)$ and $\sum_{i=1}^{n} w_i \phi_i(s)$ is also the global minimum. Linear value function approximation can represent complex value functions because the basis functions themselves can be arbitrarily complex. The goal of the reinforcement learning agent within this context is then to learn the weights w_i, which may then be used to find the policy π^* that maximises returns. We defer discussing the selection of the basis functions ϕ to the next section.

2.3.1 Learning the Value Function Online

We now focus on online learning methods. These methods process samples from the MDP individually, and update the weights of the value function at each timestep. Monte Carlo approaches to problem solving tend towards the correct answer through a large number of trials or samples. In reinforcement learning, one may learn the value function by sampling Markov chains of the MDP, and using those to update estimates of the reward at each state. In an infinite state environment, one would instead update the parameters of the value function approximation. Monte Carlo methods require a very large number of samples, and converge slowly. The Monte Carlo estimate minimises the mean squared error of the value function for the MDP in the batch case [Sutton and Barto 1998].

The value at a state may be updated using the single-step temporal difference (TD), which is the difference between the reward, and the difference between the decayed value at the successor state and the value at the current state. This is stored as the change in the value at the state,

$$\Delta V(s) = R(s) - (V(s) - \gamma V(s')),$$

used to update the value at s as

$$V(s) = V(s) + \alpha \Delta V(s),$$

where α is the learning rate. The reward in state s can be propagated further than one step. Suppose the temporal difference is now computed as

$$\Delta V(s_t) = R(s_t) - \left(V(s_t) - \sum_{\tau=1}^{n} \gamma^\tau V(s_{t+\tau}) \right).$$

Then $V(s_t)$ can be updated more accurately, but only when the state s_{t+n} is reached. TD methods are guaranteed to converge, and are the maximum likelihood estimate of the value function of the MDP when used for batch methods [Sutton and Barto 1998].

When dealing with a state-action value function $Q(s, a)$, the Sarsa method can be employed to learn and update the value function. The tuple $(s,\ a,\ R,\ s',\ a')$ (consisting of the state s and action a chosen according to policy π, the reward R, the successor state s' and action a' chosen in s') is used, giving the method its name. The change in value

$$\Delta Q(s, a) = R(s) - (Q(s, a) - \gamma Q(s', a'))$$

is used to update the value at s as

$$Q(s, a) = Q(s, a) + \alpha \Delta Q(s, a).$$

The value function Q is dependent on the policy π, making Sarsa an on-policy method. Sarsa is known to converge to Q^* provided all the state-action pairs are visited an infinite number of times (that is, the MDP is *ergodic*), and the policy π converges to the greedy policy $\pi(s) = \arg\max_a Q(s, a)$ [Sutton and Barto 1998]. When a single sample is used in the update, this is known as single-step Sarsa.

Single-step Sarsa can be augmented with eligibility traces by updating the value function as

$$Q(s, a) = Q(s, a) + \alpha \Delta Q(s, a) \Lambda_t(s, a)$$

where the eligibility trace Λ at time t is given by

$$\Lambda_t(s, a) = \begin{cases} \lambda\gamma\Lambda_{t-1}(s, a) + 1 & \text{if } s = s_t \text{ and } a = a_t \\ \lambda\gamma\Lambda_{t-1}(s, a) & \text{otherwise} \end{cases}$$

for accumulating traces, and

$$\Lambda_t(s, a) = \begin{cases} 1 & \text{if } s = s_t \text{ and } a = a_t \\ 0 & \text{if } s = s_t \text{ and } a \neq a_t \\ \lambda\gamma\Lambda_{t-1}(s, a) & \text{otherwise} \end{cases}$$

for replacing traces [Främling 2007], where λ is an additional parameter defining the trace decay. When Sarsa is employed in this way, it is referred to as Sarsa(λ). Keeping track of the trace for each state-action pair is infeasible in large state-action spaces, and impossible if either the state or action space is continuous. To that end, accumulating eligibility traces are instead stored for the approximation parameter vector $\underline{w}$, as the vector

$$\underline{\Lambda}_t(s, a) = \lambda\gamma\underline{\Lambda}_{t-1}(s, a) + \nabla_{\underline{w}} Q(s_t, a_t)$$

which for linear value function approximation reduces to

$$\underline{\Lambda}_t(s,a) = \lambda\gamma\underline{\Lambda}_{t-1}(s,a) + \underline{\phi}(s_t,a_t).$$

This is known as gradient descent Sarsa, and is the method we will primarily use in this research.

The above methods are *on-policy methods*, meaning that they learn the value function associated with the current policy, π. Off-policy methods instead learn the value function associated with a different policy regardless of how the samples are collected. Q learning [Watkins and Dayan 1992] updates the value function $Q(s,a)$ with

$$Q(s,a) = Q(s,a) + \alpha(r(s) - (Q(s,a) - \gamma \max_{a'} Q(s',a'))),$$

regardless of what action is chosen in state s' (or indeed in state s). This allows learning of the value function under the greedy policy $\pi(s) = \arg\max_a Q(s,a)$ even if the policy by which the samples are acquired is non-optimal, random, or exploratory.

Q learning is unstable for linear value function approximation [Baird 1995], which led to the creation of GQ(λ) [Maei and Sutton 2010], a generalisation of Q learning to value function approximation with eligibility traces. This was extended further as Greedy GQ [Maei *et al.* 2010]. In this algorithm, the gradient of the projected Bellman error is also tracked, as ω. A secondary learning rate β which adjusts how ω is updated is required. The linear value function approximation parameter vector $\underline{w}$ is updated as

$$\begin{aligned}
a' &= \arg\max_a Q(s,a),\\
\Delta Q &= r(s) - (Q(s,a) - \gamma Q(s',a')),\\
\underline{w} &= \underline{w} + \alpha\left\{\Delta Q \Phi(s,a) - \gamma\left((\underline{\omega}^T\Phi(s,a))\Phi(s',a')\right)\right\},\\
\underline{\omega} &= \underline{\omega} + \beta\left\{\Delta Q - \left(\underline{\omega}^T\Phi(s,a)\right)\Phi(s,a)\right\}.
\end{aligned}$$

2.3.2 Batch and Offline Learning

Whereas online methods use samples singly to learn, and then typically discard the sample, batch methods use multiple samples at a time. If one has a collection of samples and their associated rewards, one may simply construct the system giving the TD error of each sample, and find the minimising parameters of the value function approximation in the least squares sense. This is the approach taken in least-squares temporal difference learning (LSTD) [Boyan 1999]. The use of least-squares relies on the resulting system being overdetermined, meaning that there are more linearly independent samples than there are basis functions. If this is not the case, it is necessary to regularise the system, as an underdetermined system does not have a unique solution. Regularisation in reinforcement learning often takes the form of simply adding elements to the main diagonal of the system. This approach is known as ridge regression [Lagoudakis and Parr 2003] or Tikhonov regularisation and is equivalent to regularisation with an l_2 norm on the weights of the system. This means that the distance of the weights (when considered as a vector) from the origin is minimised.

Kolter and Ng [2009] employ l_1 regularisation to improve on the LSTD method through a method similar to least angle regression (LARS) called LARS-TD. This form of regularisation minimises the absolute sum of the weights. If an overdetermined system is regularised, this has the effect of altering the (already unique) answer. Regularisation increases the l_2 error of the system, although it may decrease the error in other metrics (for example, reducing test-set error by avoiding over-fitting).

LSTD-based methods produce policy-dependent solutions. If the underlying policy changes, it is necessary to gather new data. Least-squares policy iteration (LSPI)

[Lagoudakis and Parr 2003] iteratively improves the policy through repeatedly solving an LSTD system, finding an improved policy based on that solution and replaying the experience. LSPI uses a state-action value function-based variant of LSTD, LSTDQ. Sigma point policy iteration [Bowling *et al.* 2008] is a method for speeding up LSPI through a stored policy-independent experience summary, eliminating the need to store previous samples and reducing computational load.

2.4 Basis Function Schemes

A common approach to value function approximation is to select *a priori* a set of basis functions ϕ suited to general function approximation, and use a linear combination of them to represent the value function. If this set of basis functions is not modified or added to over time it is a fixed basis, whereas a basis that is modified over time is an adaptive basis. The choice of basis function can severely affect performance. We now discuss a number of common basis function schemes.

2.4.1 Tile Coding

Tile coding [Sutton and Barto 1998] is a discretisation technique where a set of piecewise constant functions is used to approximate a value function. In their simplest form, these functions act as indicator functions for an exhaustive disjoint partitioning of the states. The tiles themselves are binary, activating if a state is within that tile. A linear value function is obtained by multiplying each tile ϕ_i by a weight w_i, and summing.

The tiles can be of any shape, but typically one divides the state space in each dimension into right angled tiles of equal width. If one wishes to use a single tiling with n partitions per dimension, this results in n^d tiles of equal size across the state space.

Multiple tilings may be used, with different discretisation levels or tile shapes for each tiling, producing an overlapping set of basis functions. Typically, the different tilings are at the same discretisation level, but are offset from each other [Whiteson *et al.* 2007]. The number of tilings needed to achieve good performance grows exponentially with the number of dimensions [Wu and Meleis 2009].

2.4.2 Radial Basis Functions

Radial basis functions (RBFs) are a commonly used basis function scheme where each basis function is (in the context of reinforcement learning, usually) a multivariate Gaussian

$$\phi_i(\underline{x}) = e^{-\frac{||\underline{x}-\underline{c}_i||_2^2}{2\sigma^2}}, \tag{2.4}$$

for a given set of centres $\underline{c}$, and a variance σ^2. Sometimes, the variance itself varies in each dimension, leading to basis functions that are not strictly radial, but are stretched in some dimensions. The variances between each basis function ϕ_i may vary.

RBFs can be based around any rotated function, provided they are radial in some measure $||\cdot||$ around some centre $\underline{c}$ (that is, provided $\phi(\underline{y}) = \phi(\underline{x}-\underline{c})$ for all $||\underline{y}|| = ||\underline{x}-\underline{c}||$).

RBFs have the property that they can interpolate any function on an interval with arbitrarily high accuracy, given sufficiently many of them. They do not (strictly) form a basis, but rather form a frame. The difference between the two is discussed in section 3.2.3.

A central question for RBFs is how to set the centres and variance(s). If these are chosen in advance, a variance that is too large will lead to poor detail (but fast learning), while a variance that is too small will lead to very slow learning. Learning the centres and variances for RBFs changes the problem from linear to nonlinear, due to the exponential dependency on both parameters.

2.4.3 Polynomial Basis Functions

Polynomial functions are a natural way to represent data. By the Weierstrass approximation theorem, every function $f(x)$ continuous on an interval can be arbitrarily well approximated by a polynomial on that interval, although if interpolation is used to find that polynomial, one may run into Runge's phenomenon, where the polynomial exhibits highly oscillatory behaviour at the edges of the interval.

Lagoudakis and Parr [2003] presented a polynomial basis for value function approximation in reinforcement learning, given by

$$\phi_i(\underline{x}) = \prod_{j=1}^{d} x_j^{c_{i,j}}, \tag{2.5}$$

where $c_{i,j}$ are integers such that a tensor product of the polynomial series x^i, $i = 0, \ldots, n$ is formed.

Chebyshev polynomials of the first kind mitigate Runge's phenomenon and are more numerically stable, as their extrema are either -1 or 1, and are all found on the interval $[-1, 1]$. Chebyshev polynomials are also orthonormal. To the author's knowledge, Chebyshev polynomials have not been used in reinforcement learning to date.

2.4.4 The Fourier Basis

Fourier analysis has a rich history in science and engineering. It involves representing a function as a sum of trigonometric functions of varying frequencies. The Fourier series of a real function $f(x)$ is given by

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} \left(a_k \cos\left(k\frac{2\pi}{T}x\right) + b_k \sin\left(k\frac{2\pi}{T}x\right)\right),$$

where T is the period of the Fourier series. An approximation to the function is formed if the Fourier series is truncated. As the trigonometric functions that make up the Fourier series are mutually orthogonal, the coefficients of the series can be found as inner products, which in Euclidean space are given by

$$\begin{aligned} a_k &= \frac{2}{T}\int_0^T f(x)\cos\left(k\frac{2\pi}{T}x\right)\mathrm{d}x, \\ b_k &= \frac{2}{T}\int_0^T f(x)\sin\left(k\frac{2\pi}{T}x\right)\mathrm{d}x. \end{aligned}$$

In a signal processing context, Fourier analysis is commonly accomplished using a Fast Fourier Transform (FFT), where the coefficients of the discrete Fourier transform of a vector of length n are obtained in $O(n \log n)$.

This is fundamentally different from the approach required in value function approximation, where a function is synthesised to fit sampled data by tuning the parameters or weights of a basis set, rather than obtaining those parameters through analysis of the sampled data.

Fourier analysis has been applied to reinforcement learning in the form of the Fourier basis [Konidaris 2011]. Starting with the univariate Fourier series, the authors note that if one assumes the value function is even on the interval $[-1, 1]$, then the sine terms can be dropped from the expansion with little loss of accuracy, halving the number of terms required per dimension. A standard tensor product extension to multiple dimensions is then applied, leading to the Fourier basis,

$$\phi_i(\underline{x}) = \cos(\pi \underline{c}^i \cdot \underline{x}), \tag{2.6}$$

where $\underline{c}^i = [c^i_1, c^i_2, \ldots, c^i_d]$, $c^i_j \in [0, \ldots, n]^d$ producing a set of $(n+1)^d$ basis functions which are referred to as an nth order Fourier Basis. The Fourier Basis performs well in a number of domains when compared to other basis function schemes and will be used here as a benchmark fixed basis scheme.

2.4.5 Nonlinear Schemes

While much work in value function approximation is focused on linear schemes, nonlinear schemes are also used. Razo-Zapata *et al.* [2007] use wavelets as the activation function with a neural network, while Whiteson and Stone [2006] combined neuroevolution of augmenting topologies (NEAT) with Q learning to learn a topology and initial weighting for a neural network. Nonlinear function approximation is not often used, as it can diverge [Tsitsiklis and Van Roy 1997], and certain tasks are very difficult to solve with neural networks [Whiteson and Stone 2006]. Deep learning approaches have been applied to reinforcement learning in the form of Deep Q Learning [Mnih *et al.* 2013], which demonstrated excellent performance on Atari games.

2.5 Adaptive Basis Functions

The major drawback of fixed basis schemes is that the number of basis functions grows exponentially with the dimension of the state space; an order n polynomial basis has $(n+1)^d$ basis functions (similarly for an order n Fourier basis, or $k = n + 1$ RBFs per dimension). There are three broad approaches to this problem. Feature selection deals with this problem by selecting a sub-exponential number of basis functions from a dictionary, although often the dictionary itself is large. Constructive basis function schemes create basis functions that will best represent the value function, given some information about the domain. Finally, adaptive techniques alter the set of basis functions to balance generality with detail. We draw a distinction between these three approaches for this thesis, by specifying that feature selection refers to methods drawing from a pre-existing set of functions using batch samples, constructive techniques create those functions from batch samples, and adaptive techniques modify or add to the basis function set online. Sources in the literature may refer to all of these approaches as feature selection.

2.5.1 Feature Selection

Feature selection procedures control the size the feature set (or set of basis functions) by selecting the best features to represent the value function from a dictionary of potential features.

The OMP-BRM and OMP-TD methods of Painter-Wakefield and Parr [2012] collect samples from the MDP, and form a matrix Φ, where the element in column i, row j is the potential feature ϕ_i evaluated at $\phi_i(s_j, a_j)$. Both methods use orthogonal matching pursuit (OMP) to select a feature from the dictionary that minimises the estimated Bellman error (OMP-BRM) or TD fixed point residual (OMP-TD) at each step. This feature is then added to the set of basis functions.

The matching pursuit approach finds the set of basis functions in a dictionary with the largest projection onto the residual, which are selected for inclusion in the linear system (that is, their weights are nonzero). In orthogonal matching pursuit, the weights of all included functions are recomputed at each step through orthogonal projection onto the column space of the basis function set, while in matching pursuit, weights are assigned once only. OMP converges in the column space of the dictionary and finds a solution which approximates the l_0 pseudonorm solution [Mairal *et al.* 2008] (that is, the solution with the fewest nonzero elements, or most *sparse* solution).

The OMP-BRM method guarantees sparse recovery. That is, if the value function is m-sparse in the dictionary of candidate features (can be represented as a linear combination of m features assigned a nonzero weight), then the OMP-BRM method will recover the sparse representation in m iterations. Despite this strong guarantee which does not carry over to OMP-TD, OMP-BRM does not perform as well as OMP-TD when tested empirically. Both OMP algorithms outperform the nonadaptive method LARS-TD in a number of domains.

2.5.2 Constructive Techniques

Many techniques focus on the construction of a *best basis* for a domain based on knowledge of the domain (for example, its transition matrix or an estimate thereof). These techniques commonly assume a discrete state space, although many of these methods may be extended to continuous state spaces through the use of out-of-sample interpolation techniques. We will use the terminology of discrete state space MDPs in this section. All methods discussed below are batch methods.

Krylov bases [Petrik 2007] for discrete state spaces are formed in the context of RL by considering the first m powers of the transition matrix $\mathcal{T}$ postmultiplied by the reward vector $\underline{r}$ as basis vectors.

Proto Value Functions (PVFs) [Mahadevan 2005] are constructed by taking as basis functions the eigenvectors of the transition graph Laplacian with the *smallest* eigenvalues, as these correspond to the eigenvectors capturing the least variation. Diffusion wavelets [Mahadevan and Maggioni 2006] extend this approach by generating a wavelet basis over the graph. These two approaches require knowledge or construction of the transition graph of an MDP.

Krylov bases, PVFs and diffusion wavelets are examined in Petrik [2007] who found that Krylov methods perform better than graph Laplacian-based methods, although the application of Krylov schemes to domains with state spaces too large to enumerate might be infeasible, as the bases are as large as the state space.

Keller *et al.* [2006] use neighbourhood component analysis (NCA) to linearly project the state space in $\mathbb{R}^d$ to a much smaller space $\mathbb{R}^{\bar{d}}$, $\bar{d} \ll d$, grouping states with similar Bellman errors close together in a Euclidean sense. This projection is learned from samples of the state space, and is fed as input into a state aggregator to generate basis functions on the lower dimensional space, although the use of RBFs as basis functions is discussed. This approach relies on knowing in advance the intrinsic dimensionality of the domain, as projection to a space that is too small will result in a loss of detail. Intrinsic dimensionality could be estimated using the Minimum Description Length (MDL), the Akaike Information Criterion (AIC) [Pujol 2007], or KN rank estimation [Kritchman and Nadler 2008], provided a sample covariance matrix estimate could be computed.

Bellman Error Basis Functions (BEBFs) [Parr *et al.* 2007] take as basis functions difference between the Bellman operator applied to the value function approximation, and the value function approximation itself. This gives

$$\phi_i = R + \gamma V(s') - V(s),$$

where $V = \sum_j w_j \phi_j$. The new function ϕ_i is then added to the current representation. Of interest is the fact that this function is orthogonal to the span of V, and the sequence of functions produced in this manner are mutually orthogonal. BEBFs were extended to high dimensional state spaces by Fard *et al.* [2013] through the use of random projection techniques drawn from the field of compressive sampling. Random projection greatly reduces the dimensionality of an input signal by projecting it into a lower dimensional space through multiplication with a random matrix. Provided the signal is sufficiently sparse, it can be exactly reconstructed with high probability (it is the sparsity requirement that allows one to avoid the Nyquist limit). The BEBF random projection technique is demonstrated on a domain with six continuous state variables, discretised using 65536 overlapping tiles.

The Model Error Basis Function (MEBF) method [Parr *et al.* 2008] adds features that capture the residual error in the transition model. This paper also demonstrates that reducing the model error (defined as the error in the prediction of next feature values, and the error in the reward) results in reducing the Bellman error, as the two errors are equivalent.

2.5.3 Adaptive Techniques

Incremental Feature Dependency Discovery (IFDD) is a method for discovering dependencies between distinct binary features by estimating the *relevance* of pairwise conjunctions of the features as candidate features [Geramifard *et al.* 2011], incrementally building to the complete tensor product by adding those pairwise conjunctions with a high enough relevance to the set of basis functions. The IFDD relevance measure is the absolute value of the sum of accumulated error over the candidate function, whereas in IFDD+ [Geramifard *et al.* 2013a] the relevance measure is divided by the number of states visited where the candidate function would have a nonzero value (that is, the binary feature conjunction is activated). If a candidate feature is selected for inclusion in the feature set, the pairwise conjunctions of itself and all other features are then added to the set of candidate functions.

Batch-IFFD [Geramifard *et al.* 2013b] extended IFDD to batch learning, wherein it is shown that batch-IFDD is a matching pursuit method adding at each stage the functions with the highest correlation to the error, and thus has a convergence guarantee in the space spanned by the possible feature conjunctions.

The drawback of IFDD and its variants is that one must choose the detail level of the initial set of features. If the potential conjunctions of this set of features fails to capture an important detail, the algorithm cannot correct for that, as it has a fixed level of detail or resolution. The principle advantage of IFDD is that it avoids the curse of dimensionality by scaling polynomially with dimension.

Adaptive Tile Coding (ATC) takes the complementary approach, assuming full dimensional dependence (that is, a complete tiling), but a lack of detail in each dimension [Whiteson *et al.* 2007]. Detail is added by splitting the tiles of the initial approximation in half along the dimension that would result in the largest change to either the value function or the policy. ATC finds when to split by examining convergence. If τ (a tolerance) timesteps have gone by since the basis functions in the basis function set, $\mathbf{F}$, encountered their lowest TD error, a split is triggered. The learned weights of each potential split in each dimension are examined, and the split is performed in the tile and dimension that maximises either the change in value function, or the change in policy. The split tile is replaced by its children in $\mathbf{F}$.

Evolutionary Tile Coding (EvoTC) [Lin and Wright 2010] modifies ATC to use a genetic algorithm to decide when and where to split the tiles. Tiles may be split at arbitrary points, leading to more efficient tile representations. Whereas ATC requires specification of the threshold parameter only, EvoTC requires specification of the two mutation probabilities. Comparison between ATC and EvoTC by Lin and Wright [2010] indicates that EvoTC is more efficient with respect to tiles generated (and thus representation), but less efficient with respect to number of updates required. ATC was found to diverge on the pole balancing problem.

Adaptive RBFs [Menache *et al.* 2005] use Cross Entropy methods to tune the nonlinear parameters of a set of RBFs, allowing the functions to shift and grow or shrink, allowing for a best representation for a given number of RBFs.

2.6 Conclusion

A common policy representation for reinforcement learning problems is a value function. Value functions may be represented using a hand-picked set of basis functions, or more generally using a fixed basis. Fixed bases scale exponentially with dimension, making them difficult to apply to larger problems. Furthermore, the level of detail of a fixed basis must be chosen in advance. Adaptive basis schemes often attempt to solve one or both of these problems. Many adaptive basis schemes use tile coding, although in fixed bases there are many schemes which outperform fixed tile codings.

In the following chapter, we will introduce a fixed basis scheme which has good theoretical properties and empirical performance, and in addition lends itself well to adaptive schemes through its self-similarity.

Chapter 3

The Wavelet Basis

3.1 Introduction

Linear value function approximation involves balancing the conflicting goals of quickly learning global trends against the ability to capture dramatic changes in neighbouring states on a fine scale. The basis function schemes currently in wide use offer no good solution to this conflict.

Multiresolution Analysis (MRA) offers a framework in which multiple detail levels can be learnt at once. MRA defines a broad class of functions known as wavelets. We introduce the wavelet basis, a class of fixed basis function schemes for value function approximation which offer a number of attractive theoretical properties, and are empirically competitive against other fixed basis types.

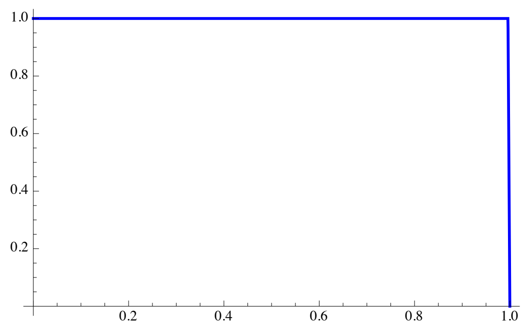
3.2 Wavelets and Their Properties

Wavelets are a comparatively new form of spectral analysis, and were developed in a number of diverse fields (leading to different viewpoints, applications and terminology). In the late 1980s, families of orthonormal compactly supported self-similar functions were found to exist. Their properties and construction are the topic of seminal work by Daubechies [1992]. In this section, we will present wavelets from a function analysis and synthesis viewpoint, bearing in mind that they will later be used for value function approximation. As wavelet analysis (like Fourier analysis) is a spectral method, the properties of wavelets are often discussed in terms of *time* and *frequency.* Time in this case corresponds directly to state space.

3.2.1 Function Approximation using Wavelets

We will begin this discussion, as much of the literature does, with the Haar wavelet. The Haar wavelet is the simplest possible wavelet, having the smallest compact support and best *time* localisation. A Haar wavelet family is comprised of a father wavelet (also known as a scaling function), and a mother wavelet (or wavelet function), shown in figure 3.1. The father wavelet defines the mother wavelet via a multiresolution analysis, which will be covered in section 3.2.2.

Approximations are formed with weighted linear sums of dilations and translations of the father and mother wavelets. A father wavelet at scale j (that is, dilated by 2^j) and at translation k is denoted $\phi_{j,k}(x) = \phi(2^j x - k)$. Similarly, a mother wavelet at scale j and translation k is denoted $\psi_{j,k}(x) = \psi(2^j x - k)$. Wavelets do not always have to be dilated by a factor of 2, but this is chosen for notational simplicity. Dilations and translations are demonstrated using B-spline wavelets in figures 3.2a and 3.2b respectively.


(a) The Haar father wavelet.

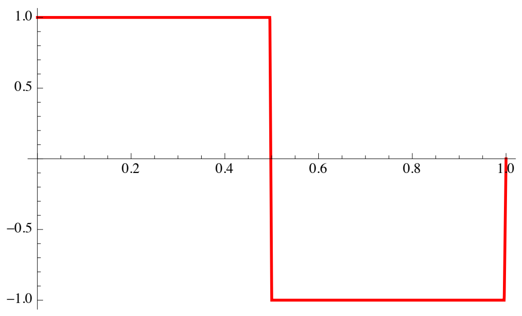
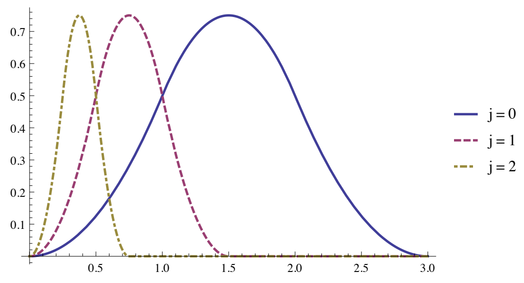
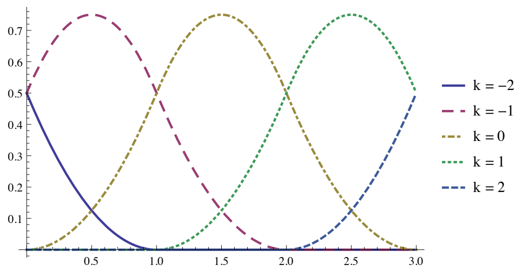

(b) The Haar mother wavelet.

Figure 3.1: The Haar wavelet family.



(a)



(b)

Figure 3.2: Three dilations (a) and five translations (b) of the second order B-spline wavelet

Suppose now we wish to approximate some function $f(x)$ using a Haar wavelet series as

$$f(x) = \sum_{i=1}^{n} w_i \phi_i(x), \tag{3.1}$$

where w_i are the weights associated with the i^{th} wavelet and $\phi_i(x)$ is either a mother or father wavelet at some location and scale. As the Haar wavelets are *orthonormal*, the inner product between any two wavelets $\langle \phi_i, \phi_j \rangle = \delta_i^j$. We can then compute the coefficients of the expansion by taking the inner product of both sides of equation 3.1 with a wavelet (say, ϕ_k).

$$\begin{aligned}
\langle \phi_k(x), f(x) \rangle &= \langle \phi_k(x), \sum_{i=1}^{n} w_i \phi_i(x) \rangle \\
&= \sum_{i=1}^{n} w_i \langle \phi_k(x), \phi_i(x) \rangle \\
&= \sum_{i=1}^{n} w_i \delta_i^k \\
&= w_k.
\end{aligned}$$

It is important to note that the set of wavelets $\phi_i(x)$ used in this expansion must be mutually orthonormal for this to work.

Father wavelets approximate a function at a specified scale. The finer the scale (that is, the higher the scale parameter), the more detailed the approximation will be. Mother wavelets add further detail to that approximation at a specific scale. A more mathematical treatment of this is given in section 3.2.2, but for now we will use the Haar wavelet to examine this intuitively.

Suppose we wish to approximate the function $P(x) = (x - 0.23)^2 + 1$ on the interval [0, 1]. If we use a single Haar father wavelet to do this, we must choose the wavelet's scale as 0. This results in the approximation shown in figure 3.3. While this is the optimal approximation, and corresponds to the function's average value across the interval, it is not a detailed approximation.

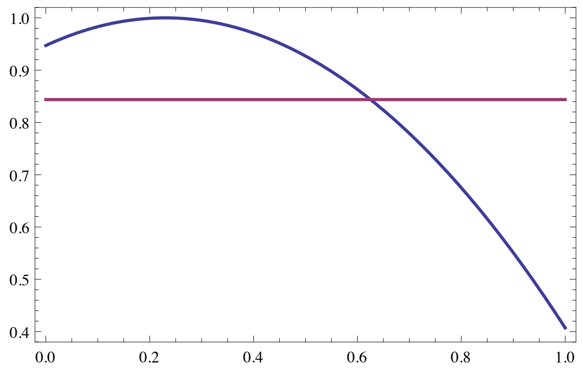


Figure 3.3: An approximation formed with a single Haar father wavelet at scale 0

We thus add a single mother wavelet function at scale 0 to the approximation, resulting in the approximation shown in figure 3.4. Further detail is added to the approximation by including the mother wavelets at scale 1 (figure 3.5a) and scale 2 (figure 3.5b). The approximation resulting from father wavelets at scale 0, and mother wavelets at scales 0 through 2 is equal to the approximation resulting from using only father wavelets at scale 3, although the functional composition of the two approximations are different. Father wavelets can be thought of as smoothing functions, while mother wavelets capture the smoothed-away detail at differing scales.

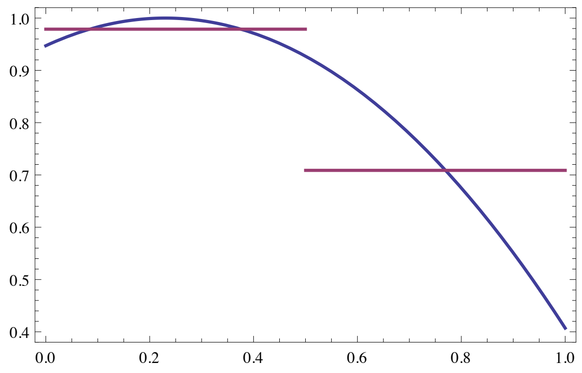


Figure 3.4: An approximation formed with a single Haar father wavelet and a single Haar mother wavelet at scale 0

Wavelets may have any number of *vanishing moments*. A wavelet has m vanishing moments if the inner product of the mother wavelet with a polynomial of degree $m + 1$ is zero. This implies that the father wavelet perfectly encodes the polynomial portion of a signal, or equivalently from a function synthesis perspective, that a combination of father wavelets can recreate any polynomial of degree m or less, with mother wavelets adding in any missing detail.

For sufficiently many vanishing moments, wavelet representations exhibit sparse coefficient representations. For any $m + 1$ times differentiable function approximated with a wavelet with m vanishing moments, the coefficients of the wavelet approximation decay

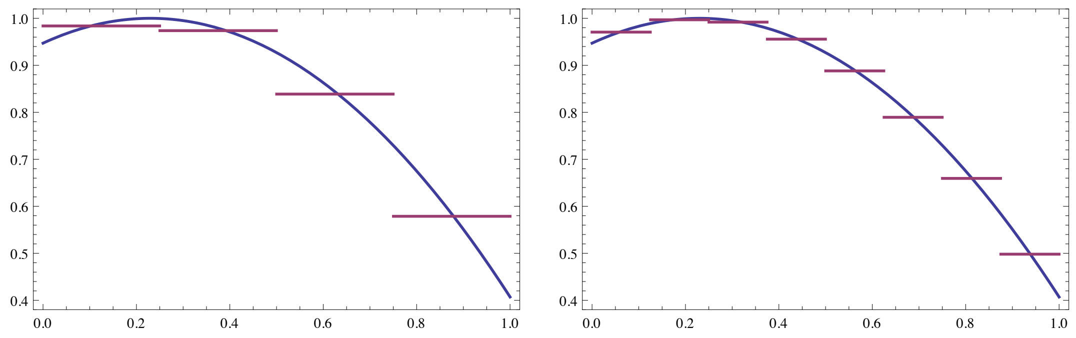

(a) Mother wavelets added up to scale 2 (b) Mother wavelets added up to scale 3

Figure 3.5: Further detail is added to the approximation using mother wavelets

exponentially with the scale [Christensen 2008], and we can thus expect an arbitrarily good approximation from a finite number of wavelets. The Haar wavelet has a single vanishing moment, and thus can perfectly represent piecewise constant functions only, forming an approximation for any other function type.

3.2.2 Multiresolution Analysis

Wavelets form a *multiresolution analysis*. A multiresolution analysis (for a dilation factor of 2) is formed when a function $f(x)$ and its translates form a basis for some function space V_0 such that $f(x) \in V_0 \Leftrightarrow f(2x) \in V_1$ and $f(2^m x) \in V_m$, and $V_0 \subset V_1 \subset \cdots \subset V_m$, $m \in \mathbb{Z}$. Subject to some eligibility criteria, such a function may be used as a scaling function $\phi(x)$ in a wavelet family, and $\lim_{m\to\infty} V_m = L_2(\mathbb{R})$. The wavelet spaces W_i are defined as the complement of the scaling function spaces, that is $W_m = V_{m+1} \ominus V_m$ and the wavelet function $\psi(2^m x)$, with its translates, forms the basis for W_m.

Thus, for any particular V_m, $\lim_{n\to\infty} V_m \oplus W_m \oplus \cdots \oplus W_{m+n} = \mathcal{L}_2(\mathbb{R})$ (defined in the following section), meaning that father wavelets form an approximation at a particular scale m, and mother wavelets add detail at that scale and finer. This allows for multiscale representations, capturing both global and local detail.

A highly abbreviated construction of a wavelet basis via multiresolution analysis is presented from Mohlenkamp and Pereyra [2008]. Suppose we find an orthogonal function $\phi(x)$ with $\langle \phi(x-j), \phi(x-k) \rangle = \delta_{j,k}$ for integers j, k, and wish to construct a wavelet basis. This function must obey a refinement equation with

$$\phi(x) = \sum_{k=-\infty}^{\infty} h_k \phi(2x - k)$$

and a finite number of h_k nonzero. In addition, we require $\int_{-\infty}^{\infty} \phi(x) \mathrm{d}x = 1$. Then, ϕ is called a scaling function or father wavelet, and one may construct a *filter* consisting of the nonzero h_k which represents ϕ. This filter may be used directly in signal processing (convolution of the filter with a discrete signal is equivalent to approximation of that signal with ϕ), or indirectly to evaluate the wavelet. In addition, we can define a mother wavelet function as

$$\psi(x) = \sum_{k=-\infty}^{\infty} g_k \phi(2x - k),$$

with

$$g_k = (-1)^{1-k} \overline{h_{1-k}},$$

where $\overline{h}$ is the complex conjugate of h.

3.2.3 Bases and Frames

In function approximation, a *basis* is a set of orthonormal functions *spanning* a function space, such that anything within that function space can be exactly reconstructed using a unique weighted sum of the basis functions. A *frame* is formed when that reconstruction is not unique, or equivalently, when the functions are not orthonormal. Frames are overcomplete, leading to redundancies in their representations (not always an undesirable property). A set of functions that is neither a basis nor a frame for a particular function space may still perform well in practice, but cannot represent all functions within the function space (for an in-depth treatment of bases and frames see Christensen [2008]). Typically, we are interested in bases or frames for $\mathcal{L}_2(\mathbb{R})$, the Hilbert space $\mathcal{H}$ of all finite square-integrable functions. In Euclidean spaces, $f(x) \in \mathcal{L}_2(\mathbb{R})$ implies that $\int_{-\infty}^{\infty} f^2(x)\mathrm{d}x$ is finite, a property satisfied by all value functions with a finite return.

The following properties of a set of orthonormal functions $\phi_i(x)$ indexed with $i \in \mathbb{N}$ are equivalent:

1. $\phi(x)$ is an orthonormal basis for $\mathcal{H}$.
2. $f(x) = \sum_i \langle \phi_i(x),\, f(x)\rangle \phi_i(x),\ \forall f \in \mathcal{H}$.
3. $\sum_{i=1}^{\infty} |\langle \phi_i(x),\, f(x)\rangle|^2 = ||f(x)^2||$.
4. $Span\{\phi(x)\} = \mathcal{H}$.

Property 3 is a special case of Parseval's inequality. For any frame,

$$c_1||f(x)^2|| \leq \sum_{i=1}^{\infty} |\langle \phi_i(x),\, f(x)\rangle|^2 \leq c_2||f(x)^2|| \tag{3.2}$$

for some constants c_1, c_2, which define the tightness of the frame. If $c_1 = c_2 = 1$, the frame is a tight frame, or basis.

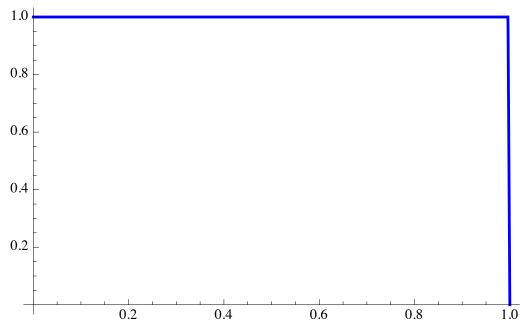
3.2.4 Wavelet Families

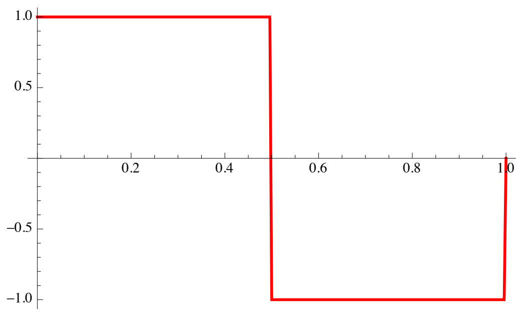
Many wavelet families exist. Here, we discuss the basic properties of a number of common wavelet families. Much of this information is drawn from Daubechies [1992] and Mohlenkamp and Pereyra [2008], but may be found in any standard wavelet reference (for example, Kaiser [2010]).

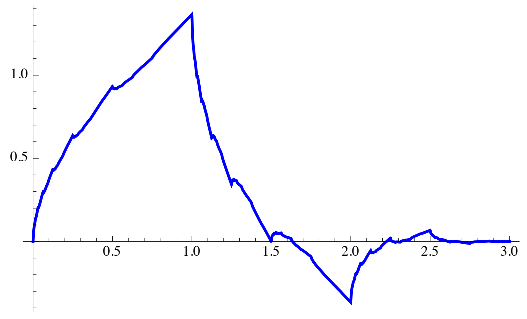
The Daubechies family of wavelets varies through a parameter value $2N$ (some texts use $N \in 2\mathbb{Z}$), defining the number of vanishing moments and the compact support of the wavelet, as well as the tap length (the number of elements in its filter). The Daubechies D_2 wavelet is the shortest possible wavelet, with one vanishing moment and a support width of one. In general, the D_{2N} wavelet has N vanishing moments, $2N$ taps (nonzero elements) in its filter, and a support width of $2N - 1$.

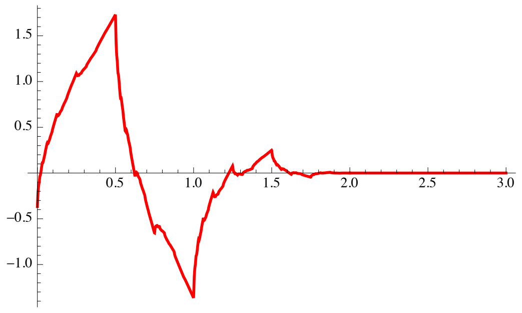
The Daubechies family has the maximal number of vanishing moments for a given support width. They are compactly supported and orthonormal, but not symmetrical. The father and mother wavelets of the first few Daubechies wavelets are shown in figure 3.6. Note that the 2 tap Daubechies wavelet is equivalent to the Haar wavelet.

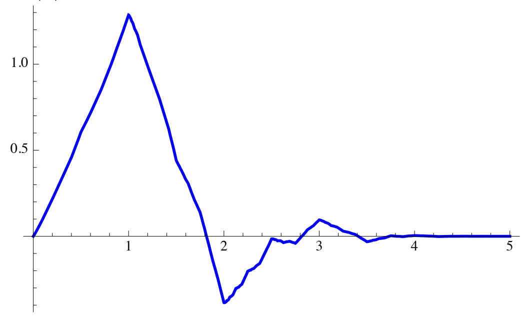
Coiflets were created so that the father wavelets have vanishing moments too. They are nearly symmetrical, and are orthonormal, but have a wide compact support. A Coiflet wavelet family is shown in figure 3.7. Symlets are a modified version of the Daubechies wavelets, which have maximal symmetry for compactly supported orthogonal wavelets through a wider compact support. They are nearly symmetric, as no wavelet may be orthogonal, compactly supported and symmetric. A Symlet wavelet family is shown in figure 3.8.

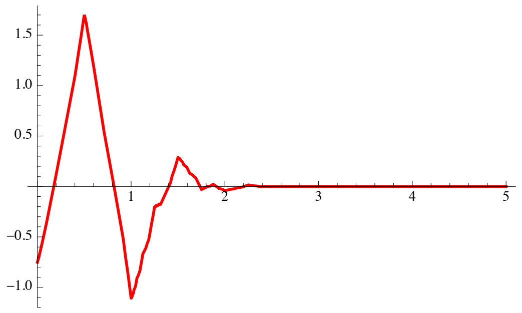

(a) The Daubechies 2 tap father wavelet

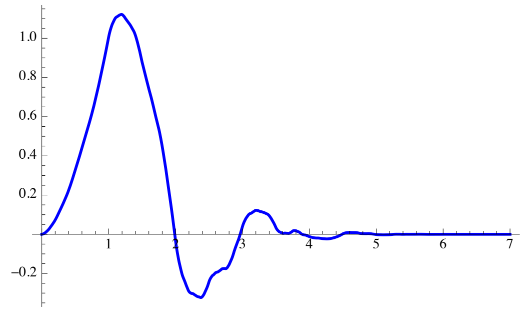

(b) The Daubechies 2 tap mother wavelet


(c) The Daubechies 4 tap father wavelet


(d) The Daubechies 4 tap mother wavelet


(e) The Daubechies 6 tap father wavelet


(f) The Daubechies 6 tap mother wavelet


(g) The Daubechies 8 tap father wavelet

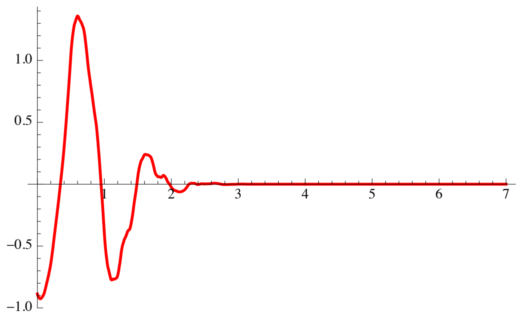
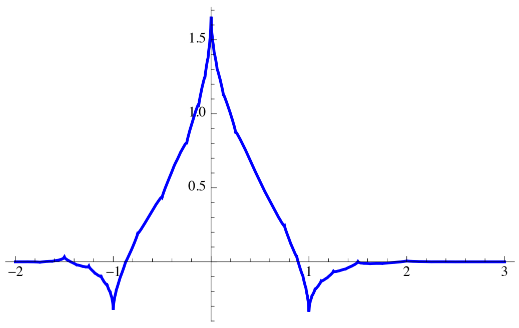
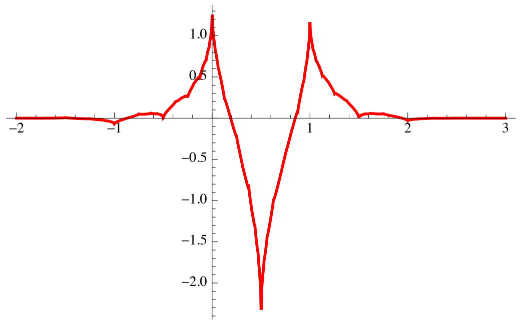

(h) The Daubechies 8 tap mother wavelet

Figure 3.6: The Daubechies wavelet family.

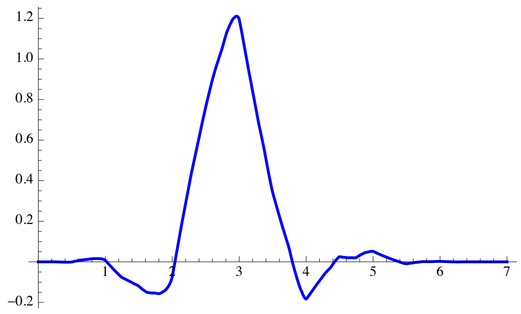


(a) The Coiflet father wavelet

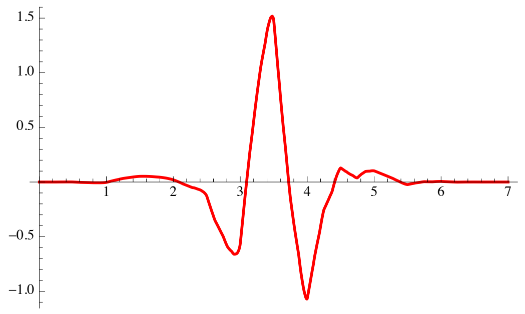


(b) The Coiflet mother wavelet

Figure 3.7: The Coiflet wavelet family



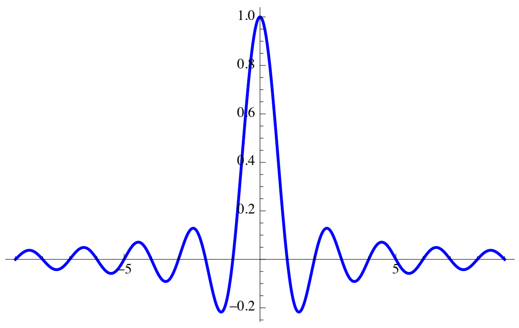
(a) The Symlet father wavelet



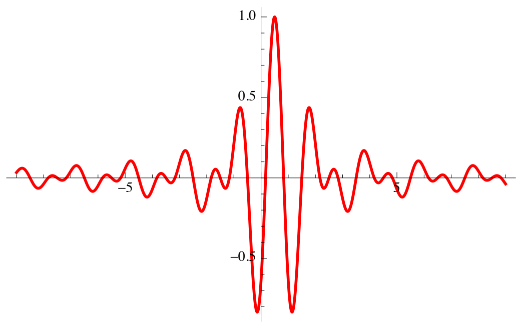
(b) The Symlet mother wavelet

Figure 3.8: The Symlet wavelet family

Shannon wavelets are orthonormal, and have a closed form expression as the sinc function. They are not, however, compactly supported. The Shannon wavelet decreases slowly, as $O(\frac{1}{x})$. They are the Fourier dual of the Haar wavelet, offering perfect localisation in frequency, but very poor localisation in time. These wavelets are shown in figure 3.9. Meyer wavelets are similar to Shannon wavelets, but sacrifice some localisation in frequency to gain exponentially rapid decay. They are shown in figure 3.10.



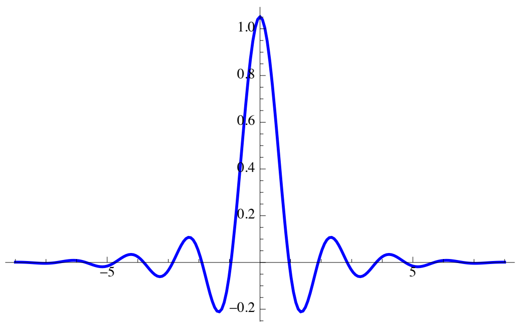
(a) The Shannon father wavelet



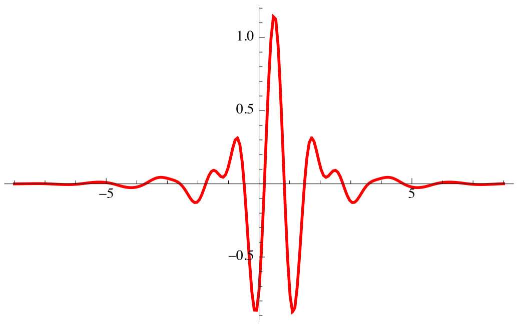
(b) The Shannon mother wavelet

Figure 3.9: The Shannon wavelet family

Biorthogonal wavelets provide good symmetry, compact support and vanishing moments, at the cost of orthonormality. Instead, these wavelets form an overcomplete frame. In order to keep the weights equivalent to an inner product a second family of synthesis wavelets orthonormal to the analysis wavelets is used. Weights are computed using the analysis wavelets, and the function can be reconstructed using the synthesis wavelets. While many biorthogonal wavelet families exist, the name usually refers to the Cohen-Daubechies-Feauveau (CDF) wavelet [Daubechies 1992]. The orders of the analysis and



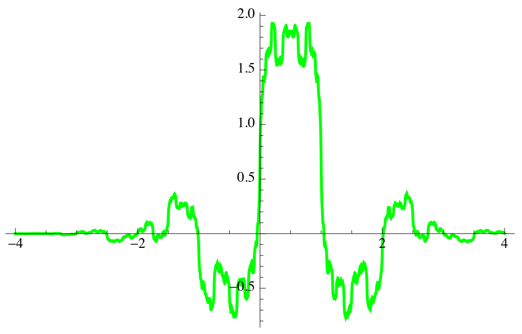
(a) The Meyer father wavelet



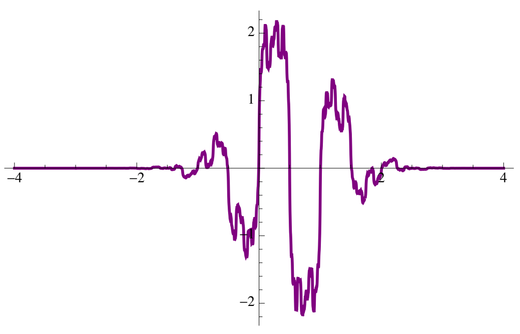
(b) The Meyer mother wavelet

Figure 3.10: The Meyer wavelet family

synthesis wavelets may differ. The analysis wavelets of a biorthogonal wavelet family are shown in figure 3.11. The synthesis wavelets, which are of interest to us for function synthesis, are shown in figure 3.12.

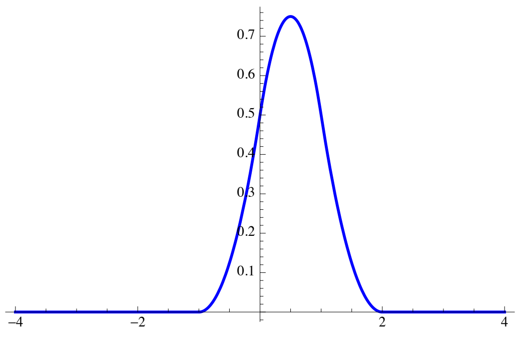


(a) A biorthogonal father wavelet

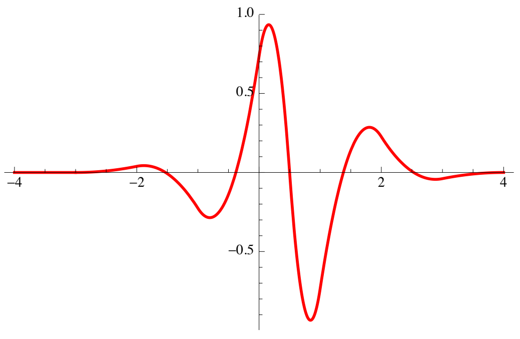


(b) A biorthogonal mother wavelet

Figure 3.11: A biorthogonal analysis wavelet family



(a) A biorthogonal father wavelet



(b) A biorthogonal mother wavelet

Figure 3.12: A biorthogonal synthesis wavelet family

Spline wavelets have received special attention in recent years due to their attractive properties for function synthesis, rather than analysis (see, for example, Unser [1997]; Cho and Lai [2005]; Unser *et al.* [1993] and references therein). Spline wavelets are constructed from splines, which can all be expressed as a combination of B-splines (or Basis splines), compactly supported piecewise polynomial functions of maximal smoothness and order for a given support width [Unser 1997].

We will discuss two such spline wavelets. B-spline wavelets are the synthesis wavelets of a biorthogonal wavelet family, and their father wavelets are given directly by B-splines shifted such that their support starts at the origin. These wavelets are not orthogonal, but are symmetric, differentiable, compactly supported and have a closed form expression. B-spline wavelets provide good localisation in time and frequency, and tend towards Gabor functions as the order of the spline tends to infinity, which have optimal localisation in both time and frequency with respect to the uncertainty principle [Unser *et al.* 1993]. These wavelets are shown in figure 3.13. Note that the zero$^{\text{th}}$ order B-spline father wavelet is a tile, or binary function.

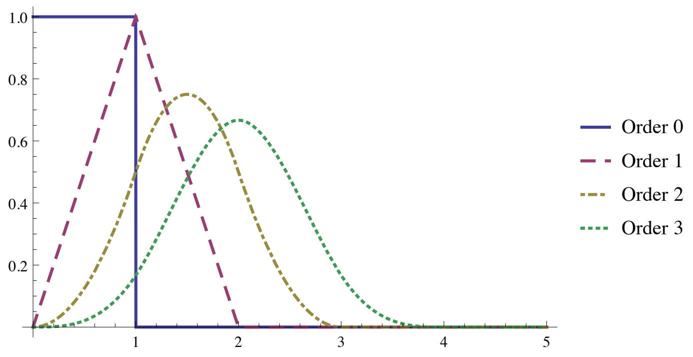


Figure 3.13: B-spline wavelets of various orders

Of all known wavelet types, B-splines will give the tightest approximation error bounds [Unser 1997]. B-spline wavelets have near optimal time-frequency localisation, even for low orders [Unser *et al.* 1993]. B-spline father wavelet functions are the smoothest possible father wavelets for a given compact support, and have the shortest support width for a given order [Unser 1997]. Furthermore, when used as interpolants, B-splines are the functions that oscillate the least [Unser 1997], resulting in high numerical stability.

The B-spline father wavelet equation is given by repeated self-convolution of a tile across the unit interval [Chui and Wang 1992]. The first three such convolutions, giving constant, linear and quadratic B-splines are

$$\phi^0(x) = \begin{cases} 1 & : 0 \leq x \leq 1 \\ 0 & : \textit{otherwise}, \end{cases}$$

$$\phi^1(x) = \begin{cases} x & : 0 \leq x \leq 1 \\ 2 - x & \;\; 1 \leq x \leq 2 \\ 0 & : \textit{otherwise}, \end{cases}$$

$$\phi^2(x) = \begin{cases} 0.5x^2 & : 0 \leq x \leq 1 \\ 0.75 - (x - 1.5)^2 & : 1 \leq x \leq 2 \\ 0.5(x - 3)^2 & : 2 \leq x \leq 3 \\ 0 & : \textit{otherwise}. \end{cases}$$

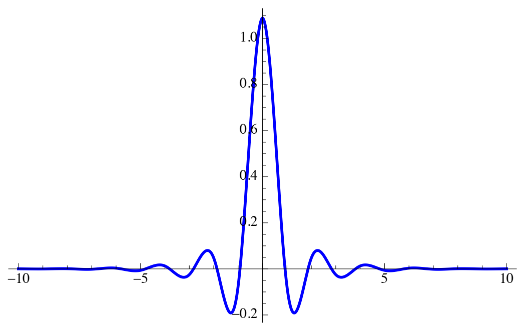
Battle-Lemarie wavelets are orthogonal spline wavelets, although the spline is in the Fourier domain, meaning that the wavelet has good frequency localisation, but infinite support (although it has exponential decay). An example of this wavelet type is shown in figure 3.14.

Family	Order	Support Width	Orthonormal	Symmetric	Closed Form
Haar	-	1	Yes	Yes	Yes
Daubechies	$2n$	$2n-1$	Yes	No	No
Coiflet	$2n$	$3n-1$	Yes	No	No
Symlet	$2n$	$2n$	Yes	Nearly	No
Shannon	-	$\mathbb{R}$	Yes	Yes	Yes
Meyer	-	$\mathbb{R}$	Yes	Yes	No
B-spline	n	$n+1$	No	Yes	Yes
Battle-Lemarie	n	$\mathbb{R}$	Yes	Yes	No

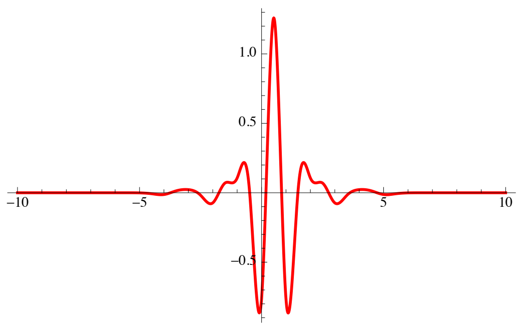
Table 3.1: Comparative properties of wavelet families

Family	Special Properties
Haar	Best spatial localisation, worst frequency localisation
Daubechies	Maximal vanishing moments for support width
Coiflet	Additional vanishing moments in mother wavelets
Symlet	Least asymmetry in a compactly supported orthonormal wavelet
Shannon	Best frequency localisation, worst spatial localisation
Meyer	Closed form in Fourier domain and exponential decay across $\mathbb{R}$
B-spline	Tends towards optimal time-frequency localisation
Battle-Lemarie	Closed form in Fourier domain and exponential decay across $\mathbb{R}$

Table 3.2: Unique features of wavelet families



(a) A Battle-Lemarie father wavelet



(b) A Battle-Lemarie mother wavelet

Figure 3.14: A Battle-Lemarie wavelet family

The properties of these wavelet families are summarised in table 3.1, with the unique properties of each wavelet summarised in table 3.2. This list is by no means exhaustive, as wavelet families can be constructed to have any particular set of properties in mind, provided one bears in mind the trade-offs between support width, symmetry and other properties.

All the wavelets examined thus far map from $\mathbb{R}$ to $\mathbb{R}$. Multiwavelets extend the univariate construction to wavelets mapping from $\mathbb{R}^n$ to $\mathbb{R}$ by considering construction through dilation matrices (for example, dilation by $2^j I$). Further details about these wavelets can be found in Strela [1996] or Keinert [2003].

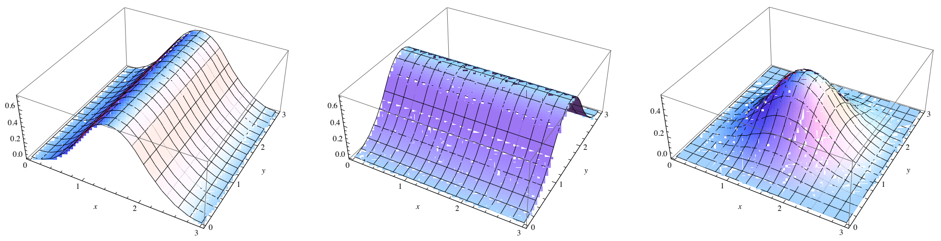
Coifman and Maggioni [2006] introduced diffusion wavelets, a new construction generalising wavelets from Euclidean spaces to graphs and manifolds. Discussion of the application of diffusion wavelets to reinforcement learning [Mahadevan and Maggioni 2006] is deferred to section 3.6.

3.3 The Wavelet Basis for Reinforcement Learning

We now examine a generalised wavelet basis for use in reinforcement learning. We assume function approximation in a continuous state space rescaled to be on the unit hypercube, but any finite interval will also work. We will denote the collection of nth order father wavelet functions $\phi(x)$ for representation of a function $f(x)$ on the support [0, 1] at scale j as $\phi_{jk}(x) = \phi(2^j x - k)$ for $k = -n+2, \ldots, 2^j - 1$, $j, k \in \mathbb{Z}$ and similarly, the set of mother wavelet functions $\psi(x)$ at scale j are given as $\psi_{jk}(x) = \psi(2^j x - k)$, with finer scale sets taking on increasing values of j. Through the multiresolution property discussed in section 3.2.2, the combination of father wavelets $\phi_{jk}(x)$ for some specific scale j, and mother wavelets $\psi_{\bar{j}k}$ for $\bar{j} \geq j$ forms a basis for $\mathcal{L}_2(\mathbb{R})$.

3.3.1 Extension to Multiple Dimensions

In order to extend univariate wavelets to $\mathbb{R}^d$, we take the *tensor product* of the wavelets in each dimension as described by Chui [1997] and Triebel [2008]. In practice, this is simply the outer product of the basis functions in different dimensions at each scale. Orthogonality between wavelets in different dimensions is maintained, as is regularity. As the wavelets are of compact support in each dimension, their products have compact support across all dimensions. This has the effect of producing what we will refer to as a wavelet *tiling*, as shown in figure 3.15. By combining the wavelets across all the dimensions, we form a basis for $\mathcal{L}_2(\mathbb{R}^D)$ [Triebel 2008].



(a) A father wavelet in x only, constant in y

(b) A father wavelet in y only, constant in x

(c) The product of father wavelets in x and y

Figure 3.15: A father wavelet tile formed in x and y

Let ψ^G denote either a father wavelet function $\psi^S = \phi$ or a mother wavelet function $\psi^W = \psi$. For a given scale j, we form an initial approximation to the function using father wavelet tiles

$$\Phi_{j,\vec{k}} = \Psi^G_{j,\vec{k}} = \prod_{d=1}^{D} \psi^{G_d}(2^j x_d - k_d),$$

where $\vec{k} = [k_1, \ldots, k_D] \in \mathbb{Z}^D$ are the offsets in each dimension, and for all dimensions d, $G_d = S$. By varying $\vec{k}$ we include all tiles such that the wavelet tile is nonzero in the region of interest.[1] Further detail is added with *cross tiles* $\Psi^G_{\bar{j},\vec{k}}$ created from all possible products of at least one mother wavelet function and father wavelet functions ($\exists d\, G_d = W$) at scales $\bar{j} \geq j$. It is important to note that the cross tiles do not include tiles comprised of father wavelets only, as that makes a father wavelet tile.

3.3.2 Parameters

The scale of the wavelet approximation is an obvious parameter that must be chosen. Fortunately, this scale needn't be chosen *ad hoc*. For Daubechies wavelets, an n-tap

[1]In value function approximation, the region of interest is usually the unit hypercube of scaled states.

wavelet at scale j has a support width of $\frac{n-1}{2^j}$. Since we are interested in approximating a function on the unit interval [0, 1], it would make sense to choose the largest j such that the support width of the wavelet is not less than the unit interval, as this is the final scale at which the wavelet functions are global in scale across that interval. Additional detail added with wavelet functions will be on a local scale, not covering the unit interval entirely. Similar choices can be made for other wavelet families: the scale should be chosen such that the compact support of the father wavelets is approximately the unit interval.

For the Daubechies wavelets, we must decide what tap length to use. This influences the smoothness of the resulting function, as well as its support width. Daubechies wavelets offer the highest number of vanishing moments for a fixed support width. This means that a Daubechies $2n$-tap wavelet will have a support width of $2n-1$, and n vanishing moments, giving exact synthesis of polynomials of order $n-1$ and below, provided sufficiently many wavelets are used to entirely cover the support of that polynomial. The trade-off between vanishing moments and support width can be summarised as follows: for each additional vanishing moment (that is, additional order of polynomial smoothness in representability) one must include two extra functions due to the support width increasing by two.

B-Spline wavelets have the polynomial order of the splines as their parameter, with $n \in \mathbb{N}_0$. The support width of the B-spline father wavelet is $n+1$. B-Splines are symmetric, but sacrifice orthonormality (and are thus overcomplete, forming a frame). They are the synthesis wavelets of the biorthogonal wavelet family. They offer exact order n polynomial reconstruction across any fixed interval.

The locations of each wavelet are fixed by the wavelet scheme, an advantage over RBFs. There is only one possible wavelet transform, given a starting scale.

The wavelet basis (as it has been introduced thus far) lacks a method for selecting the number of basis functions per dimension, which is possible in other basis function schemes. As one must choose the wavelet family and order, and the starting scale, which combined determine the number of wavelets per dimension, one may ask if this may be done in reverse: choose the family and number of wavelets per dimension, and through that determine the scale and order of the wavelets. This may be done by fixing one of the free parameters. If the wavelet order is fixed, then one may select an initial scale such that the number of wavelets generated is close to the number of wavelets requested (rarely will these match up exactly).

3.3.3 Wavelet Families for Reinforcement Learning

There are an infinite number of possible wavelet families, but we will focus on three with properties that may be desireable in reinforcement learning. We defer demonstrating their application to section 3.5, and instead concentrate here on explaining the use of these functions in function approximation.

The first type, Haar wavelets, are binary functions across the interval [0, 1], which produce value functions equivalent to those produced by a disjoint regular single tile coding, although Haar wavelets offer a number of advantages over and above tile coding. Like tile coding, Haar wavelets have a simple closed-form expression that can be rapidly evaluated. Haar wavelets form the lowest order of many wavelet families. The Daubechies 2-tap wavelet is a Haar wavelet, as is the B-spline of order 0.

Haar wavelets are symmetric and orthonormal, but cannot represent functions that are not piecewise constant. Higher-order polynomials can be represented with a B-spline wavelet transform, or a Daubechies wavelet transform, among others. B-spline wavelets are the symmetric synthesis wavelets of the biorthogonal wavelet transform, and as such sacrifice orthonormality for symmetry. They have simple closed-form expressions as polynomial splines of order n across a support of $n+1$. The order may be chosen through balancing the number of intrinsic wavelets in the transform against the desired piecewise

Family	Width	Orthonormal	Symmetric	Closed Form	Order
Haar	1	Yes	Yes	Yes	0
Daubechies 4-tap	3	Yes	No	No	1
Daubechies 6-tap	5	Yes	No	No	2
Daubechies 8-tap	7	Yes	No	No	3
B-Spline order 1	2	No	Yes	Yes	1
B-Spline order 2	3	No	Yes	Yes	2
B-Spline order 3	4	No	Yes	Yes	3
B-Spline order 4	5	No	Yes	Yes	4

Table 3.3: Properties of wavelet families suggested for use in RL

polynomial smoothness. As is the case with polynomial bases, it is seldom worthwhile to consider orders higher than 4. B-Splines form a frame, as they are not orthogonal. This leads to overcomplete, nonunique representations, which is sometimes advantageous.

Daubechies wavelets are highly asymmetric and do not have closed-form expressions, but they have the maximum number of vanishing moments of all orthogonal wavelets, leading to the highest level of unique polynomial representation and a very sparse representation. The lack of a closed form expression means wavelets must be evaluated iteratively. The Daubechies-Lagarias algorithm can be used to evaluate a wavelet function at a point [Soman *et al.* 2009] for a specified level of accuracy in constant time—similar to the evaluation of sine and cosine, which are computed through series approximations. The wavelet tiles can thus be evaluated in time linear to the number of dimensions.[2]

To summarise, if one wishes to use a binary function representation, Haar wavelets offer a number of advantages over tile coding. If one desires unique representations and orthonormality, Daubechies wavelets offer the most sparse representation. If one prefers symmetry, B-spline wavelets can be quickly evaluated and have a closed form expression. These properties are summarised in table 3.3 for suggested orders of these wavelets. Note that in this table order is taken to mean the order of piecewise polynomial that the family can represent perfectly.

3.4 Wavelets on the Interval

The interval across which wavelets are placed can be arbitrarily rescaled, provided wavelets are placed with respect to this rescaling. As an example, if one wanted three Haar wavelets at scale 0 on the unit interval, one could rescale that interval to be [0, 3], or equivalently rescale the wavelets (and their translations and scaling) to be the desired size.

Daubechies wavelets with compact support of length greater than one have a problem when used to approximate functions on the unit interval. We define a region of interest (ROI) to be the real segment [0, 1]. The Haar wavelet has a support width of exactly one, leading to perfect localisation in time (and very poor localisation in frequency). If one chooses to use this wavelet, and selects a scale j at which to add father wavelets, then exactly 2^j father wavelets will be used, all of which will be fully within the ROI. This is a consequence of each father wavelet at scale j having a support width of $\frac{1}{2^j}$ from the dilation equation.

All other wavelets, however, have a larger support width (say, [0, n]). Under the wavelet equation

$$\phi_k^j(x) = 2^{\frac{j}{2}}\phi(2^j x - k),$$

[2]We used the Java Wavelet implementation within the mathIT package (`http://www.math-it.org/`).

each father wavelet ϕ has a support width of $\frac{n}{2^j}$ and is shifted by $\frac{k}{2^j}$. A complete wavelet basis is formed by every integer k such that at least part of ϕ is within the ROI, for some scale j. As an example, consider the D_8 Daubechies wavelets: these wavelets have a support width of 7, and so at scale 2 there are 10 wavelets within the ROI, 4 induced by the 2^2 positive shifts of k. The long 'tail' of this wavelet family results in 6 seemingly extraneous 'tail' wavelets, regardless of the scale chosen.

These tail wavelets contribute little to the approximation when Daubechies wavelets are used, and are present at every scale, and in every wavelet family (aside from Haar); the number of tail wavelets is one less than the support width (or in the case of wavelets without compact support, the effective support width). For a wavelet family with (effective) support width n, and scale m, the number of wavelets per dimension is $2^m + n - 1$. As the total number of wavelets scales exponentially with the number of dimensions in a fixed basis, these tail wavelets pose a serious problem, as they lead to an exponentially large number of wavelets with poor representational capability.

Depending on the wavelet family, one may choose to exclude these tail wavelets from the approximation (thus sacrificing some representational ability). One may also construct a wavelet family on the interval through multiresolution such that all the wavelets have compact support entirely within the interval, although the form and compact support of the wavelets at the interval edges will be different to the other wavelets, whether constructed to supplement an existing wavelet family, or constructed entirely on the interval.

Alternatively, the interval can be repeated infinitely, effectively wrapping the wavelet transform toroidally around the region of interest (also called periodising the wavelet). Repeating the interval requires that the interval end-points of the function match, otherwise one will induce 'ringing', or pseudo-Gibbs phenomena at the interval edges. In the context of function synthesis, one may choose to introduce a gap in between the repeated intervals, wherein the wavelets may take on arbitrary values to reduce this effect. Intervals may also be mirrored to mitigate this effect (also called wavelet folding in the literature), but this doubles the length of the interval to approximate. We shall investigate simple periodisation with gaps within this chapter through toroidal Daubechies wavelets at scale 0, compressed such that they operate over the unit interval. For in-depth discussions of the other forms of wavelets on an interval, see Mohlenkamp and Pereyra [2008].

3.5 Empirical Results

3.5.1 Methodology

We test B-spline, Daubechies, tailless Daubechies and toroidal Daubechies wavelet bases against the Fourier basis, RBFs and tile coding in five online domains to demonstrate that wavelets are competitive with commonly used fixed basis schemes. Tail-less wavelets were selected such that only those tiles with more than 90% of their energy within the domain of interest were retained. Toroidal wavelets employed Daubechies wavelets of order n at scale 0 (having support width $n - 1$) where the wavelet was resized and periodised to wrap around the unit interval, with a small gap (of precisely one unit before resizing). A non-overlapping single tile coding was used, with equal numbers of tiles used in each dimension.

Parameters are varied to show the performance of 4, 6, and 8 basis functions per dimension, as shown in table 3.4. We used Sarsa(λ), with $\lambda = 0.9$, $\epsilon = 0.05$ and $\gamma = 1$. PARL2 α scaling is used [Dabney 2014] with $\alpha_0 = 1$. Results are averaged across 100 experiments for 20 episodes (following the structure of Konidaris [2011]), to examine the early performance of each basis function type. All results show the average return measured at each episode. Functions which diverged, or performed an order of magnitude worse than the best performing function for all episodes are not shown to make the graph

Wavelet	Order	Scale	Functions
B-spline	2	1	4
	2	2	6
	4	2	8
Daubechies	6	1	6
	8	1	8
Tailless Daubechies	8	1	4
	8	2	6
Toroidal Daubechies	4	0	4
	6	0	6
	8	0	8

Table 3.4: Number of functions for the tested parameter combinations

Domain	Dimensions	Actions
Discontinuous Room	2	4
Mountain Car	2	3
Acrobot	4	3
Pinball	4	5
3D Mountain Car	4	5

Table 3.5: Dimensions and actions of each tested domain

scales readable. The number of dimensions and actions of each tested domain are shown in table 3.5.

3.5.2 Discontinuous Room

Discontinuous room is a continuous state gridworld with a narrow gap to the goal room [Konidaris *et al.* 2011]. Curiously, every tested basis function exhibited a performance dip on the first episode, with the exception of tiling (figures 3.16, 3.17 and 3.18). RBFs did not manage to solve this task, perhaps due to the narrowness of the gap. The B-spline basis outperforms all other bases for all tested levels of detail.

3.5.3 Mountain Car

The mountain car task Sutton and Barto [1998] has a steep discontinuity in the value function. The B-spline wavelet basis performs well in this domain, as shown in figures 3.19, 3.20 and 3.21, outperforming the other basis function types.

3.5.4 Acrobot

The acrobot [Sutton and Barto 1998] has 4 continuous variables (an angle and an angular velocity for each joint). The second order B-spline wavelet basis at scale 2 outperforms the other basis functions tested (figure 3.23), but is outperformed by the O(3) and O(7) Fourier basis (figures 3.22 and 3.24). Both RBFs and tiling perform poorly in this domain.

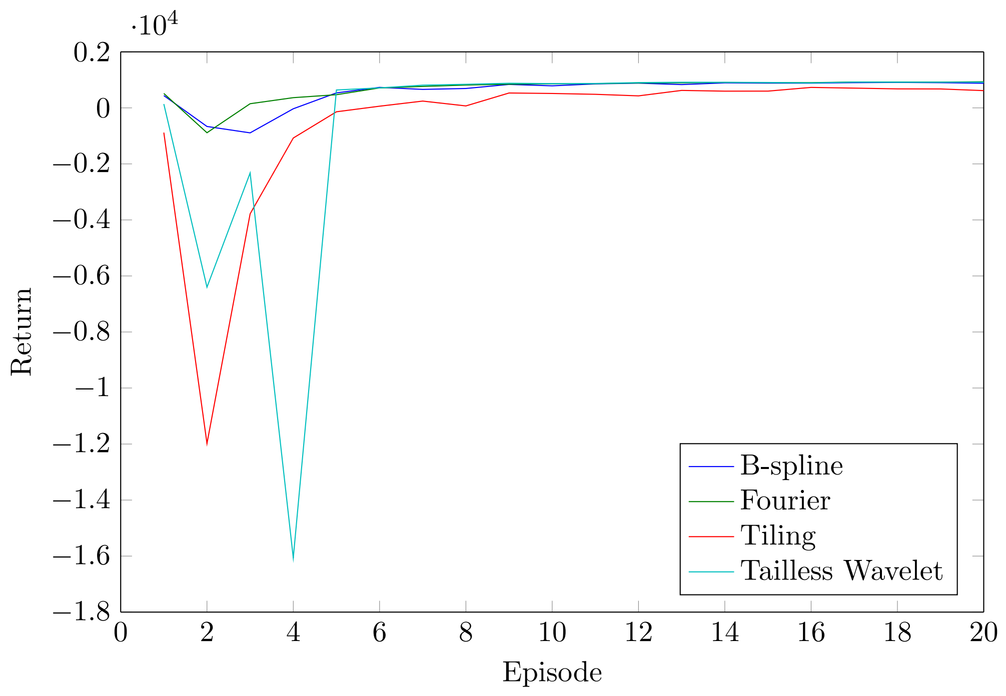


Figure 3.16: Discontinuous room returns using 4 basis functions per dimension

3.5.5 Pinball

In the pinball task [Konidaris and Barto 2009] the agent must control a ball with elastic dynamics through a two dimensional obstacle field. A tiling comprised of 4 tiles per dimension performs surprisingly well against O(3) Fourier (figure 3.25), but these results do not carry over to greater levels of detail, wherein the B-spline basis outperforms the others (figures 3.26 and 3.27). RBFs did not converge for this domain, and are not shown. The pinball map used is shown in figure 3.28.

3.5.6 3D Mountain Car

3D mountain car [Taylor *et al.* 2008] is an extension of the mountain car task into a three dimensional valley. RBFs perform well in this domain, equalling the performance of the B-spline wavelet basis (figures 3.29, 3.30 and 3.31). Both outperform other basis function schemes in 3D mountain car. Tiling performs poorly.

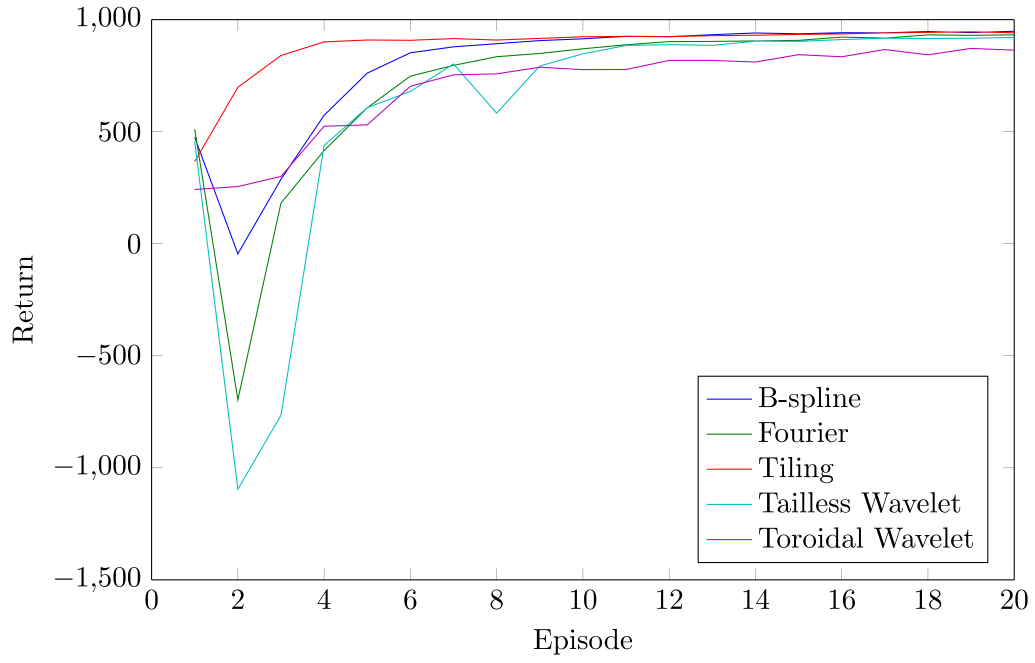


Figure 3.17: Discontinuous room returns using 6 basis functions per dimension

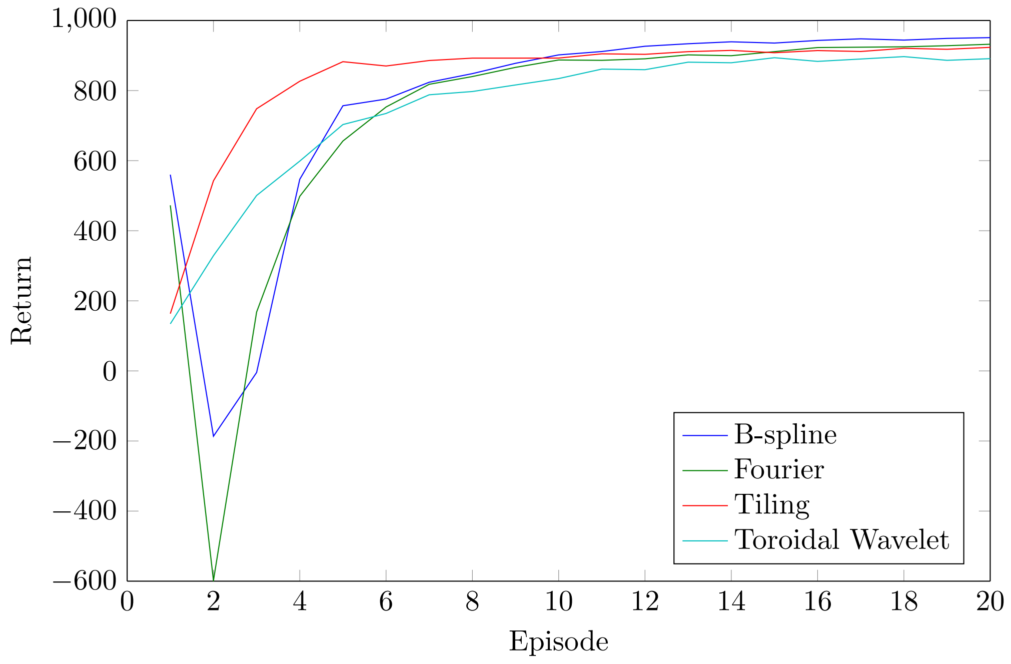


Figure 3.18: Discontinuous room returns using 8 basis functions per dimension

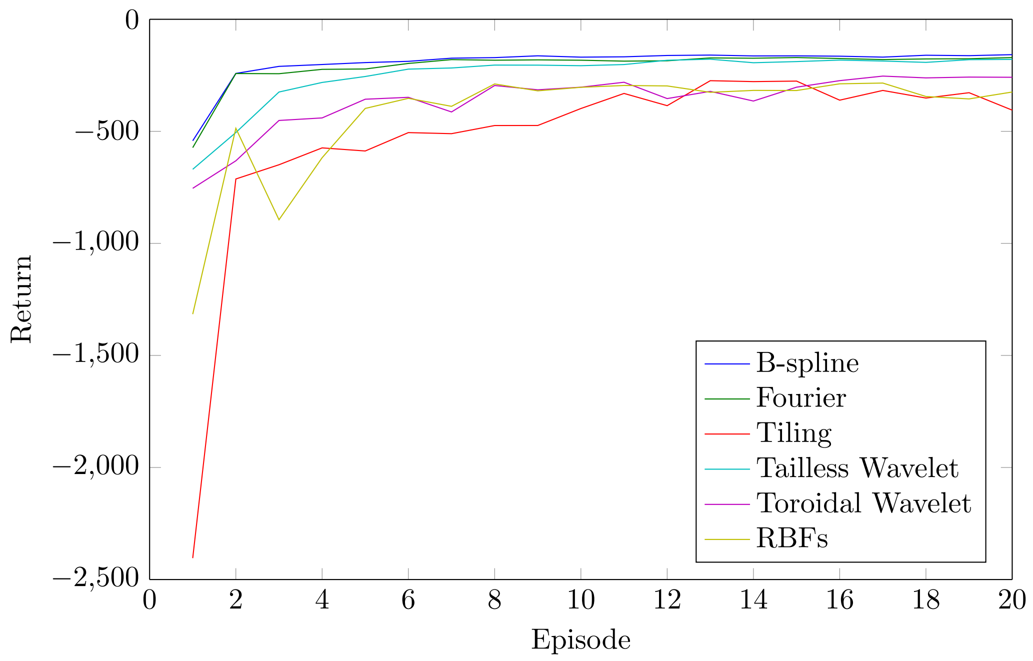


Figure 3.19: Mountain Car returns using 4 basis functions per dimension

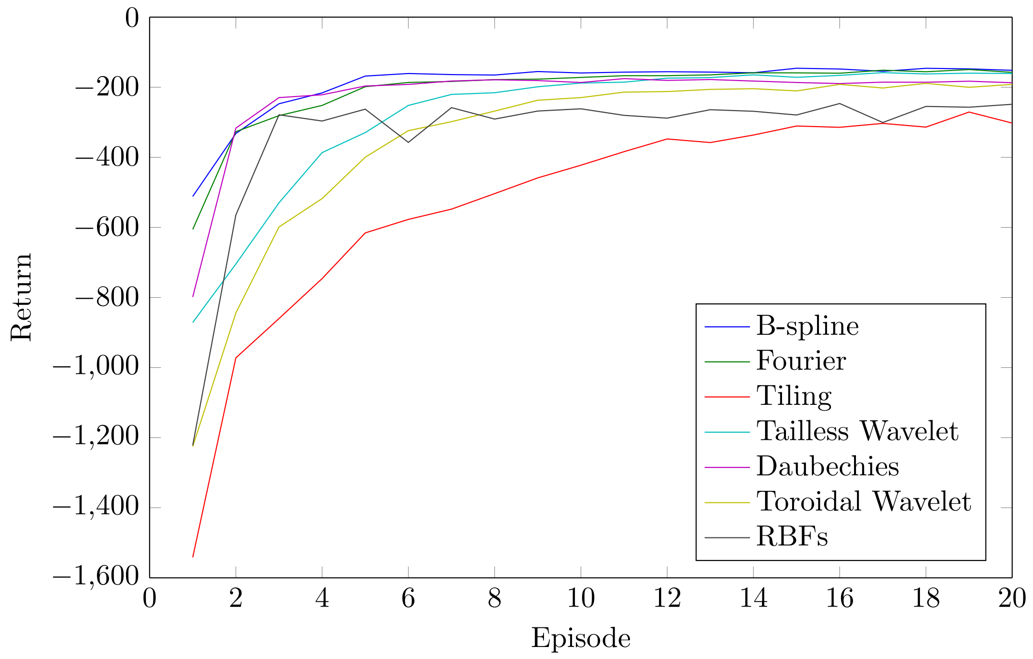


Figure 3.20: Mountain Car returns using 6 basis functions per dimension

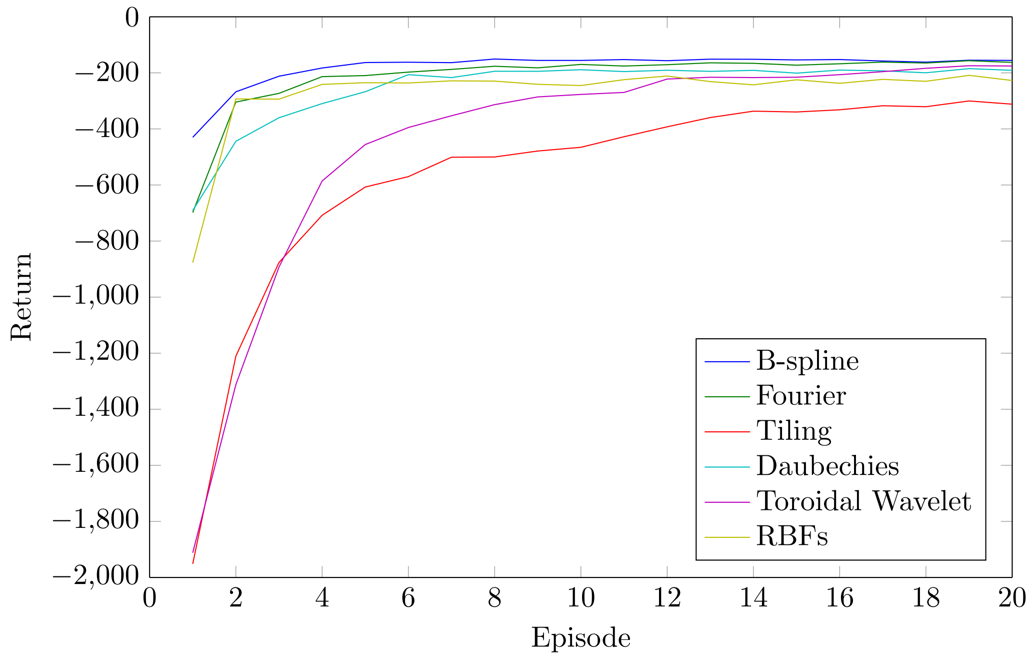


Figure 3.21: Mountain Car returns using 8 basis functions per dimension

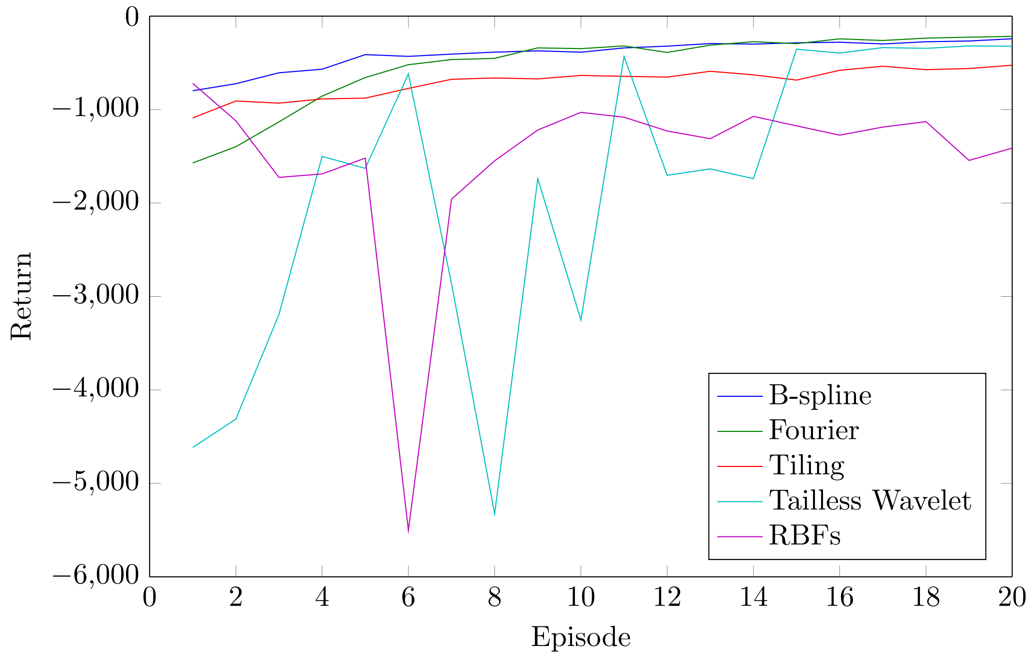


Figure 3.22: Acrobot returns using 4 basis functions per dimension

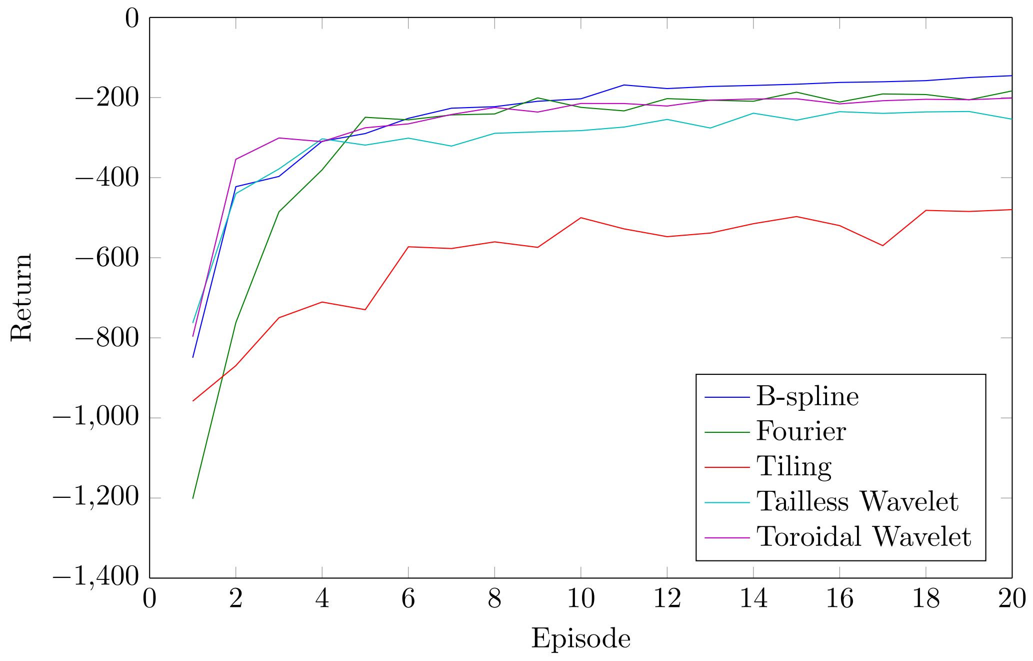


Figure 3.23: Acrobot returns using 6 basis functions per dimension

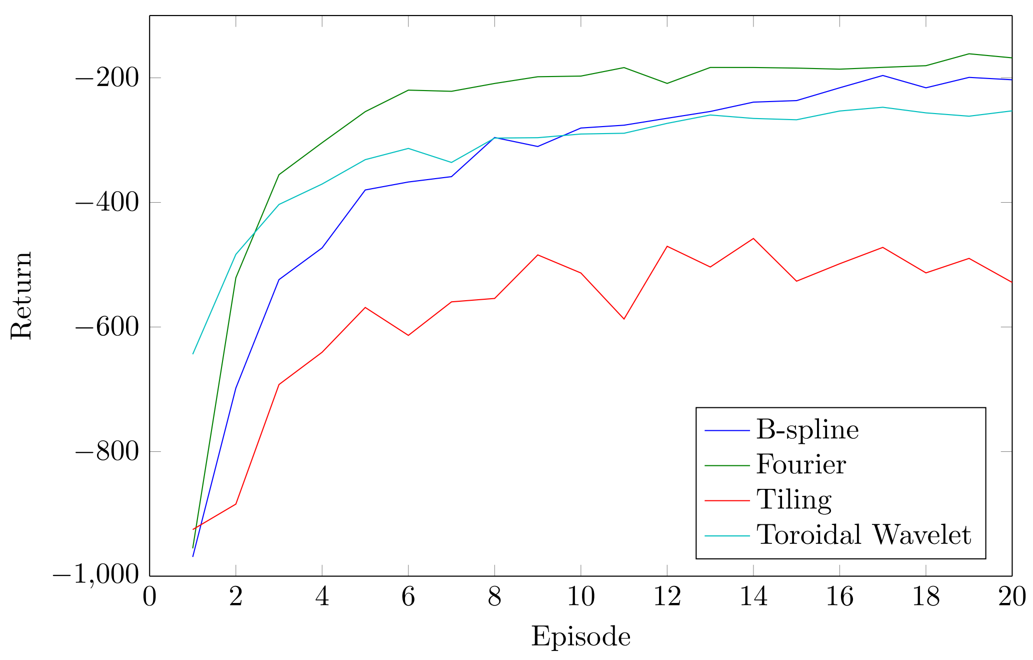


Figure 3.24: Acrobot returns using 8 basis functions per dimension

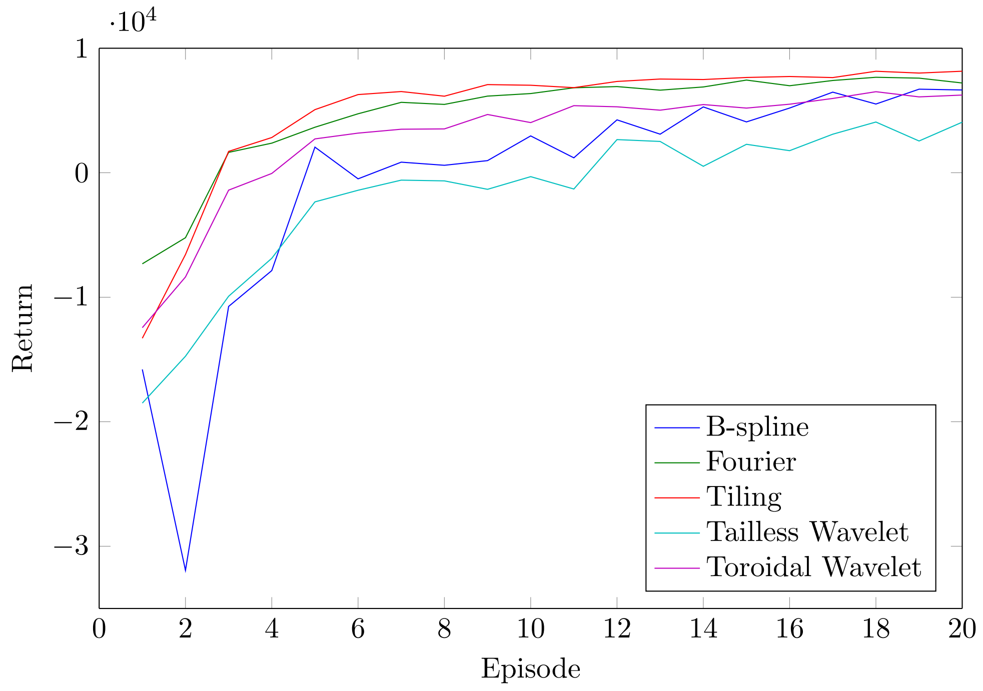


Figure 3.25: Pinball returns using 4 basis functions per dimension

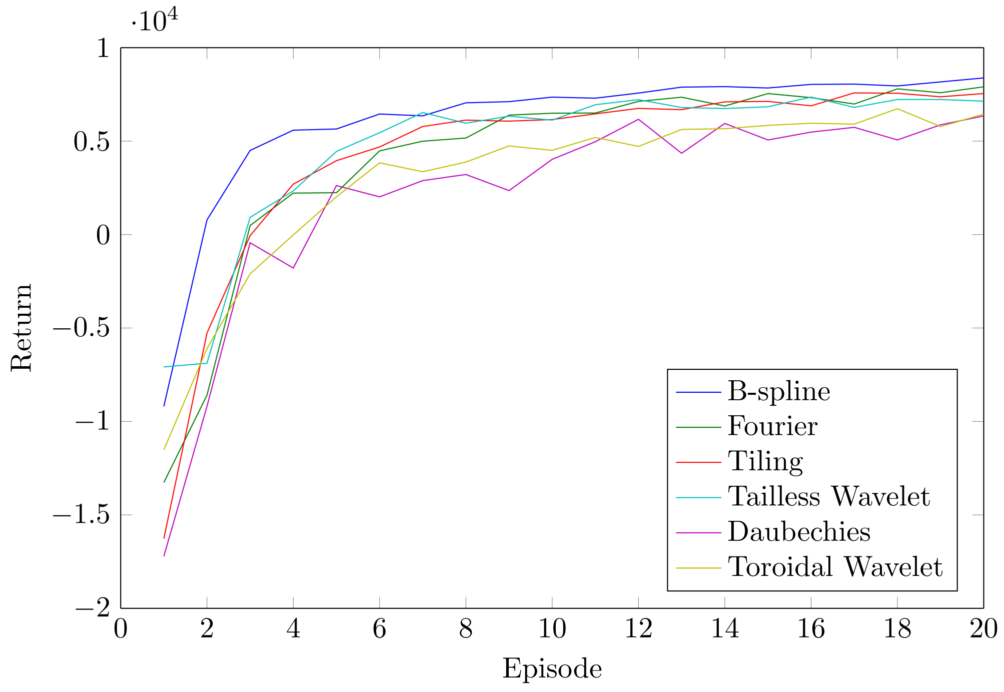


Figure 3.26: Pinball returns using 6 basis functions per dimension

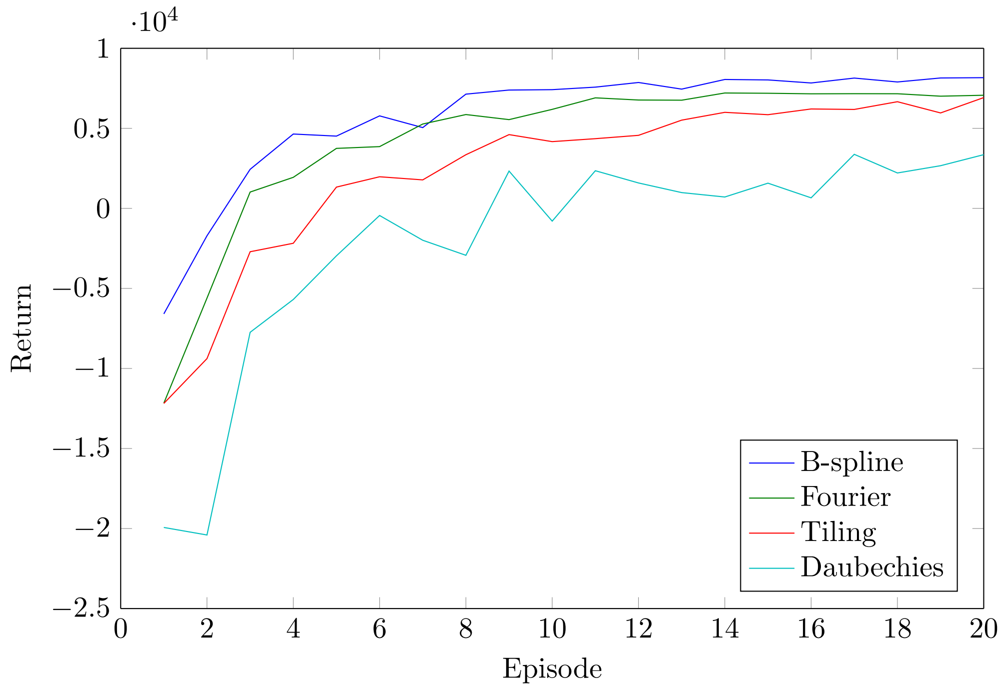


Figure 3.27: Pinball returns using 8 basis functions per dimension

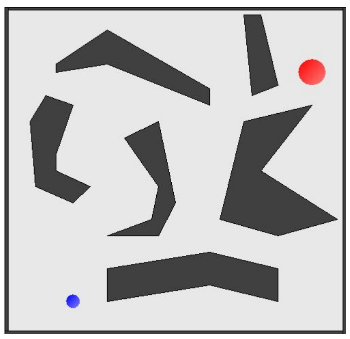


Figure 3.28: The pinball map used

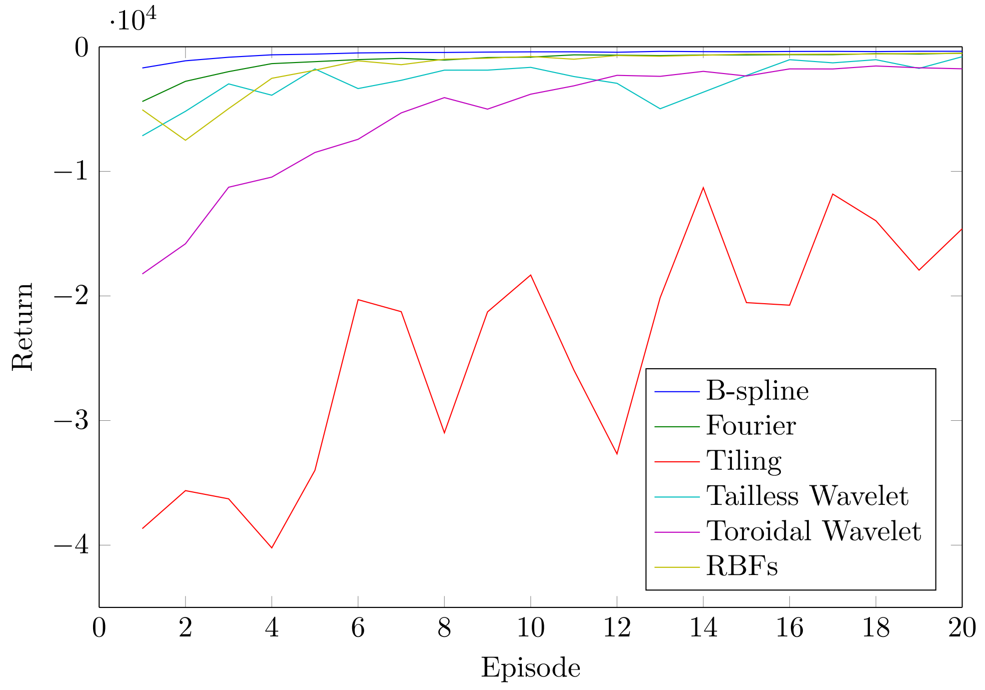


Figure 3.29: 3D Mountain car returns using 4 basis functions per dimension

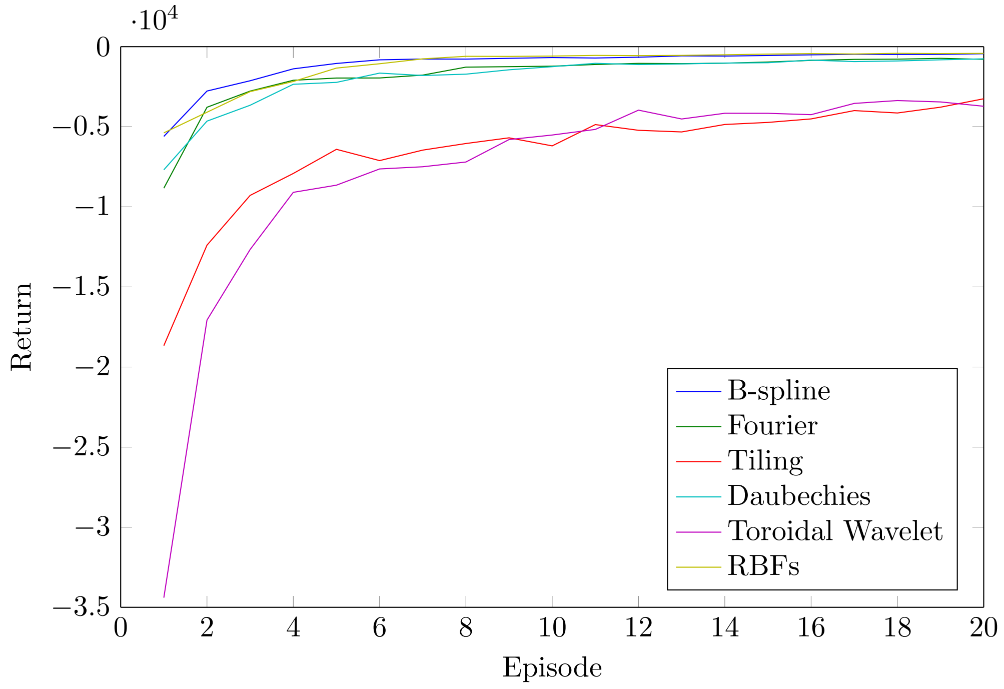


Figure 3.30: 3D Mountain car returns using 6 basis functions per dimension

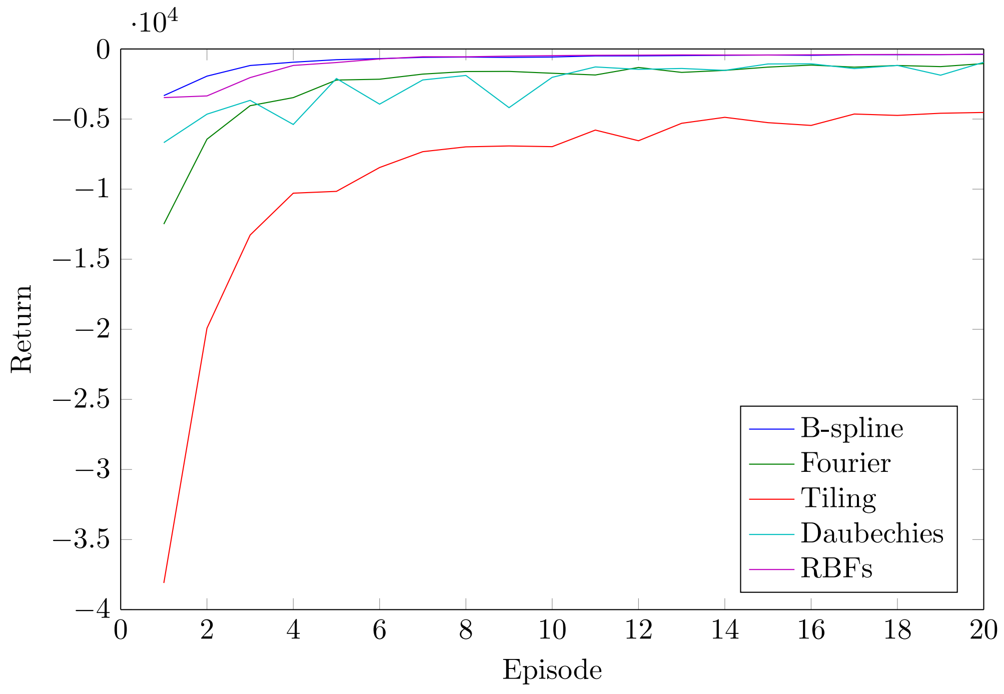


Figure 3.31: 3D Mountain car returns using 8 basis functions per dimension

	Fourier	RBFs	Tiling	B-splines
Mountain Car	**53**	279	284	62
Discont. Rm.	188	-	**53**	209
Pinball	5634	-	4529	**3598**
Acrobot	194	2810	351	**121**
3d Mtn. Car	1320	1253	4080	**645**

Table 3.6: Average standard deviation for each basis function type and domain.

	Fourier	RBFs	Tiling	B-splines
Mountain Car	94.25	95.31	99.51	**93.41**
Discont. Rm.	99.93	99.95	99.97	**99.92**
Pinball	443.29	448.01	445.49	**442.21**
Acrobot	**82.14**	94.46	95.62	87.38
3d Mtn. Car	**79.85**	91.25	98.16	81.56

Table 3.7: LSTD residuals for each basis function type and domain.

3.5.7 Further Results

The standard deviations (measured across the runs and averaged across the episodes for six basis functions per dimension) for selected basis function types and all domains are shown in table 3.6. Of interest is that the standard deviation of the B-spline basis is by far the lowest for pinball, acrobot and 3d mountain car. Tile coding has the lowest standard deviation for discontinuous room, and the Fourier basis the lowest for mountain car.

Six basis functions per dimension were used in an LSTD method Boyan [1999] (regularised with ridge regression with $\delta = 0.01$) using 10 000 samples collected in 50 state chains by a random policy. The Euclidean norms of the residuals are shown in table 3.7, indicating that the use of B-splines (in mountain car, discontinuous room and pinball) and the Fourier basis (in acrobot and 3d mountain car) resulted the smallest approximation error.

3.6 Discussion and Conclusion

The fixed B-spline wavelet basis is shown to be competitive against other basis function schemes in both online and batch settings, due to its polynomial representational sparsity. Daubechies wavelets do not perform as well as the B-spline wavelets, due to their large compact supports. While excluding the wavelet 'tails' mitigates this problem, the wavelets still do not have the low-order flexibility seen in the B-spline wavelet type. Toroidally wrapped Daubechies wavelets perform overall poorly. We will therefore carry the B-spline wavelet basis forward as the exemplar wavelet basis.

The wavelet basis is not only useful in its own right, but it provides a framework on which one may build a basis with good theoretical guarantees. Vanishing moments provide a characterisation of the polynomial representational sparsity of the basis, while compact support is necessary for approximation with a finite number of basis functions.

Wavelet approaches have been used in reinforcement learning in only a few instances. Razo-Zapata *et al.* [2007] use wavelet networks for function approximation in continuous state spaces, by combining wavelet and neural networks. This work made use of the Mexican hat wavelet, a continuous, non-orthogonal wavelet equal to the derivative of a Gaussian. Wavelet contributions are thresholded to avoid problems with the infinite support width of this wavelet. Ganesan *et al.* [2007] use wavelet-based denoising to clean incoming data in a reinforcement learning problem, thresholding the coefficients of a wavelet transform.

Mahadevan and Maggioni [2006] use diffusion wavelet techniques to construct basis functions by constructing a graph of the MDP and generating a diffusion wavelet tree over it. These wavelets are constructed by modelling the diffusion of information through a graph or manifold. Although diffusion wavelets are more general than standard wavelets, using this basis requires either a model of the transition manifold of the MDP or a sufficient number of transition samples to build an approximate graph. It also requires an out-of-sample extension method when encountering states that are not already on the graph. Direct comparison of these two methods is not feasible due to the vastly different problem domains addressed, as the wavelet basis assumes a continuous Euclidean state space with an unknown transition function.

The wavelet basis presented in this chapter is a fixed basis, and thus still suffers from the central weakness of any fixed basis: the curse of dimensionality. The number of basis functions needed to cover a finite space is exponential in the number of dimensions of that space. In order to mitigate this problem, we will need to add basis functions adaptively.

Chapter 4

Relevance Measures

4.1 Introduction

While the wavelet basis performs well for the demonstrated domains, one may ask the same question as with any fixed basis: is this the best possible basis for a specific problem? The schemes discussed in sections 2.5.1 and 2.5.2 have the advantage of selecting or constructing a best basis using *a priori* information about the problem, such as the transition model or sparsity, or having a large number of samples with which to assess the worth of a candidate function. In an online setting, the usefulness, or *relevance* of a function must be estimated at the same time as learning the value function to which the function may contribute by estimating the correlation of the function with error. In this chapter, we examine two different approaches in the literature, and define a number of novel relevance and error measures which are based on estimates of function projections.

Adaptive tile coding (ATC) [Whiteson *et al.* 2007] directly computes the relevance of candidate basis functions using TD(0). Those functions which would maximally change either the value function or the policy through their inclusion in the basis function set are assigned a high relevance. This approach requires one to track the weights of all potential features using the same learning mechanism as those of the basis function set. As candidate basis functions may replace others in the basis function set, this would become complicated were one to use Sarsa(λ), for example, as one would need to track weights and eligibility traces for each potential function in the assumed absence of the function they would replace. One may also ask if maximal change is necessarily the correct goal.

Incremental feature dependency discovery (IFDD) [Geramifard *et al.* 2011] is an online approximation of a matching pursuit algorithm, using a relevance measure to identify which conjunction of binary features to include in the basis function set. For a particular binary feature ϕ, its relevance for T samples is defined as

$$\rho(\phi) = \sum_{t=0}^{T} |\delta(s_t)|,$$

using the TD error $\delta(s)$ for all s_t within the domain of ϕ, a divergent measure of the accumulated absolute error across ϕ. Subsequently, IFDD+ [Geramifard *et al.* 2013a] used

$$\rho(\phi) = \frac{|\sum_{t=0}^{T} \delta(s_t)|}{\sqrt{T}},$$

for all s_t within the domain of ϕ, which is a divergent measure of the absolute average error across ϕ multiplied by the square root of the number of samples comprising this estimate. Note that the absolute value is now applied to the sum, rather than each term, the primary difference between the two algorithms. In IFDD, a perfectly approximated value function may still accumulate a high relevance due to stochasticity in the TD error, which we believe to indicate a typographical mistake. The intended usage of both relevance measures is clear: if a candidate function has a high error across its domain, then adding it to the basis function set will reduce that error. The drawback of these relevance measures comes when you consider their divergence. Since both either grow infinitely unless the TD error δ vanishes, they do not take into account what one would consider an acceptable level of error. This is problematic for continuous-state MDPs, where the value function approximation might never be exact. Furthermore, if one wanted to use a value function approximation scheme that was nonbinary such as RBFs, the Fourier basis, the polynomial basis, or the wavelet basis introduced in the previous chapter, there is no clear way of computing the function relevance.

4.2 Novel Relevance Measure

We note that a compactly supported function ϕ extended to multiple dimensions has a domain Ω outside of which the function is identically zero. The size of this domain, $\|\Omega\|$, which can be computed as the product of the support widths across all dimensions. We define a per-function relevance $\rho(\phi)$ at time T with domain Ω of size $\|\Omega\|$ as

$$\rho(\phi) = \|\Omega\| \frac{\sum_{t=0}^{T} \delta(s_t)\phi(s_t)}{T}, \tag{4.1}$$

for all states s_t within Ω, where $\delta(s)$ is the TD error in state s, and function ϕ is assumed to be normal (although not necessarily orthonormal). To facilitate updates, we will store the relevance ρ and the number of samples T used to create it separately. When a new sample is gathered (say, at s_{T+1}), we update ρ using

$$\rho(\phi) \leftarrow \frac{T}{T+1}\rho(\phi) + \frac{1}{T+1} \|\Omega\| \, \delta(s_{T+1})\phi(s_{T+1}).$$

We note that $\frac{\sum_{t=0}^{T} \delta(s_t)\phi(s_t)}{T}$ is an *average.* This relevance measure is convergent, rather than the divergent measures of IFDD and IFDD+; it is an online estimate of the correlation between δ and ϕ.

Theorem 4.1. *The relevance measure ρ in equation 4.1 is a Monte Carlo approximation of the inner product between the function ϕ and the Bellman error Δ across all states with respect to the state density.*

Proof. Define the Bellman error function $\Delta(s) = \mathbb{E}\left[\delta(s)\right]$ assuming an MDP with policy inducing ergodic chains. Suppose the weights w are fixed and learning of the relevances has converged such that ρ is known for all basis functions ϕ. Then, since each state has been visited an infinite number of times,

$$\lim_{T\to\infty} \|\Omega\| \frac{\sum_{t=0}^{T} \delta(s_t)\phi(s_t)}{T} = \int_{\mathcal{S}} \Delta(s)\phi(s)P(s)ds, \tag{4.2}$$

where $P(s)$ is the state density function and $\mathcal{S}$ is the set of all states, as

$$\lim_{T\to\infty} \frac{\sum_{t=0}^{T} \delta(s_t)\phi(s_t)}{T}$$

is the expected mean value of $\Delta(s)\phi(s)P(s)$. This integral is equivalent to the inner product $\langle\Delta, \phi\rangle_P$, which is a projection of ϕ onto Δ. If a finite number of samples are drawn, the relevance measure is an estimate of this quantity. □

Recall that Ω is the domain of ϕ, and so outside of this domain, the inner product of ϕ with any function is identically zero.

For optimal policies and value functions, the expected value of the relevance is zero. Intuitively, this is because, while there may be representational or stochastic error at any particular timestep, the basis functions comprising the value function must be the best possible fit. This implies that the expected projection onto the error must be zero. We prove this

Corollary 4.1. *For a fixed optimal value function with a policy π^* inducing ergodic Markov chains, the expectation of the per-function relevance given in equation 4.1 is zero in the limit as time tends to infinity, for all basis functions within the basis function set.*

Proof. Suppose the relevance is not zero in the limit for some function. This implies that there is a function ϕ whose expected inner product with the error δ is not zero (since we have taken the limit that time tends to infinity, the relevance is this inner product). This implies that the function has a nonzero projection onto the error, and thus the error could be reduced by changing the function's weight. This violates the assumption that the value function is optimal. Thus, the expected relevance is zero for all functions. □

Suppose now that we compare the relevances of two different functions. We would prefer that the relevance is constructed using many samples, rather than few. The relevance measure used in IFDD+ [Geramifard *et al.* 2013ab] assigns a higher relevance when more samples are used by dividing the error sum by the *square root* of the number of samples, which is equivalent to multiplying the average error by the square root of the number of samples; that is,

$$\frac{\sum_{t=0}^{T} \delta(s_t)}{\sqrt{T}} = \sqrt{T}\frac{\sum_{t=0}^{T} \delta(s_t)}{T},$$

which is equivalent to $\sqrt{T}\rho(\phi)$ if ϕ is binary and all samples are within its domain. We could employ a similar technique for ρ, but this would make our relevance divergent, rather than being an inner product estimate. Instead, we will multiply the relevance by $\frac{T-1}{T}$ to get

$$\rho(\phi) = \frac{T-1}{T} \left\|\Omega\right\| \frac{\sum_{t=0}^{T} \delta(s_t)\phi(s_t)}{T}, \tag{4.3}$$

which attenuates relevances that have few samples, but in the limit of T does not affect any of its properties. In principle, one could attenuate the function more with a numerator of the form $\max\{0, T-n\}$ for any $n \geq 1$, but we choose conservative attenuation.

4.3 Weighted Relevance Measure

Suppose now we consider our relevance measure under a changing policy π, tending towards π^* via a learning mechanism. Our relevance will no longer converge to zero, as the initial samples will involve errors from a different policy. To mitigate this, we will use an exponentially weighted moving average (EWMA) instead of the average given by

$$\frac{\sum_{t=0}^{T} \delta(s_t)\phi(s_t)}{T}.$$

Given a weighting ϵ, the update formula for the EWMA E_t of a data series d_t is given by

$$E_{t+1} = \epsilon E_t + (1-\epsilon)d_{t+1}.$$

The previous estimate E_t is decayed by ϵ, in a manner similar to the decay seen in the return and the eligibility trace. Note that E_0 is undefined. A common estimate for the initial mean is $E_0 = d_0$ (since, for a single sample, the sample itself is the mean), but this can bias the EWMA heavily towards the initial sample. Consider $\epsilon = 0.99$. After updating E_{100}, d_{100} has a weight of 0.01, while d_0 has a weight of 0.366, the highest of any sample. It will take 459 samples for the weight assigned to the initial sample d_0 to decay below that of the most recent sample.

For our purposes, we require the weights to monotonically increase. To this end, we will choose the initial estimate to be $E_0 = (1-\epsilon)d_0$, such that the first sample is multiplied by the same factor as every other sample. This is equivalent to choosing $E_0 = 0$ if the data series starts at d_1. E is now an underestimate of the EWMA, tending to it as the number of samples grows.

The EWMA is an estimate of the mean of a process [Čisar and Čisar 2011]. Using the sample-based attenuation of the previous section, the EWMA-based relevance measure is thus given by

$$\rho(\phi) = \frac{T-1}{T} \|\Omega\| (1-\epsilon) \sum_{t=0}^{T} \epsilon^{(T-t)} \delta(s_t)\phi(s_t), \tag{4.4}$$

with the update formula

$$\rho(\phi) \leftarrow \frac{T}{T+1}(\epsilon\rho(\phi) + \|\Omega\| (1-\epsilon)\delta(s_{T+1})\phi(s_{T+1})),$$

necessitating storage of ρ and T only. This exponentially weighted relevance measure provides us with a number of attractive theoretical properties.

Theorem 4.2. *The EWMA-based relevance measure of equation 4.4 is a Monte Carlo approximation of the inner product between the function ϕ and the Bellman error δ across all states with respect to the state density, provided the learning mechanism converges to a policy π^* with value function Q^*_π.*

Proof. In the limit,

$$\lim_{T\to\infty} \frac{T-1}{T}(1-\epsilon) \sum_{t=0}^{T} \epsilon^{T-t}\delta(s_t)\phi(s_t) = \lim_{T\to\infty} (1-\epsilon) \sum_{t=0}^{T} \epsilon^{T-t}\delta(s_t)\phi(s_t),$$

which is the expected average of $(\phi(s)\delta(s)P(s))$ (since this is just the EWMA in the limit) across the state space for policy π^*, as samples drawn from previous policies are exponentially discounted. When this is multiplied by $\|\Omega\|$, it is equivalent to $\int_{\mathcal{S}} \Delta(s)\phi(s)P(s)ds$. □

The expectation of the weighted relevance given in equation 4.4 is zero in the limit as time tends to infinity, provided the learning mechanism converges to a policy π^* with value function Q^*_π.

Lemma 4.1. *Suppose that some candidate function can be expressed as a linear combination of functions already within the basis function set, that is*

$$\phi = \sum_i w_i\phi_i.$$

Then, the relevance of ϕ will tend to zero provided the learning mechanism converges for all w_i.

Proof. Since we know that $\phi = \sum_i w_i\phi_i$, we can express the inner product as $\langle\phi, \delta\rangle_P = \sum_i w_i\langle\phi_i, \delta\rangle_P$. If each $\langle\phi_i, \delta\rangle_P = \rho(\phi_i)$ tends to zero, the linear combination must also tend to zero. □

This means that a candidate function which is redundant through representation in the basis set will have a relevance that tends to zero.

4.4 Measuring Error

It is useful to also define related measures that measure error in a meaningful way, using similar techniques. These measures do not converge to zero in the limit, but instead tell us something about how much error can be attributed to a specific function.

We define the *absolute error* $A(\phi)$ of a function ϕ as

$$A(\phi) = \|\Omega\| \frac{\sum_{t=0}^{T} |\delta(s_t)|}{T}, \tag{4.5}$$

with extensions as per the relevance to sample-based attenuation and exponential weighting. This error measure is simply the average absolute error observed over Ω, the domain of ϕ. It converges (under either a fixed policy, or variable policy with exponential weighting) to the expected absolute error across Ω (one need only consider the theorems above with ϕ identically one).

We define the *observed error* $O(\phi)$ of a function ϕ as

$$O(\phi) = \|\Omega\| \frac{\sum_{t=0}^{T} |\delta(s_t)| \, \phi(s_t)}{T}, \tag{4.6}$$

with extensions as per the relevance to sample-based attenuation and exponential weighting. This provides a measure of the absolute error that the function may contribute to. If one considers a ϕ with some subdomain identically zero, this function may have a high absolute error measure across that domain, but no change in function weight will affect this error. As per the theorems above, this error measure will converge to the inner product $\langle|\delta|\phi\rangle$, a finite quantity expressing the error across Ω that ϕ may alter.

4.5 Measuring the Relevance of Functions

For *candidate* functions which do not yet form part of the set of basis functions that determine the value function, the relevance is a direct measure of the usefulness of that function; a candidate function with a high relevance has a large estimated projection onto the error function $\Delta(s)$, while a function with a low relevance either has a low projection onto the error due to a poor fit, or due to a low error.

Functions forming part of the basis set are slightly more difficult. The relevance for such a function will tend towards zero regardless of how well or poorly the function fits, and the absolute error and observed error will converge to some finite measure, even if the weight for that function has converged. One may need to consider a measure involving both. A high absolute or observed error, with a low relevance, may mean that the function is a poor fit. One may choose to measure the ratio between the absolute or observed error and the relevance, for example. As the relevance tends to zero, while the error measures tend to some finite number, the ratio will tend to infinity unless the error tends to zero.

A more stable measure is the difference between the observed error and the relevance. This will tend to the average absolute error weighted by the function value (as the relevance tends to zero), and will always be non-negative if the basis functions used are non-negative.

4.6 Empirical Results

The relevance and error measures presented in this chapter have interesting theoretical properties. To further demonstrate the use of these properties, we will show their correlation with function weights empirically. We stress that these empirical results are largely qualitative: there are few points of possible comparison. We ran a fixed domain and extracted the weights assigned to the basis functions, along with each function's relevance, absolute error, observed error and the difference between the observed and absolute errors and the relevance, each with and without decay, and with and without sample-based attenuation.

The domain chosen is mountain car, which provides a representationally complex value function with a low dimensionality, allowing for many more experiments than larger domains. The weights of the functions were sorted by magnitude, and plotted in figure 4.1 for 1, 20 and 1000 episodes. Relevance and error measures are shown in figure 4.2. The results for episodes 1 to 20 were averaged across 1000 experiments, while the results for episode 1000 were computed separately and averaged across 10 experiments.

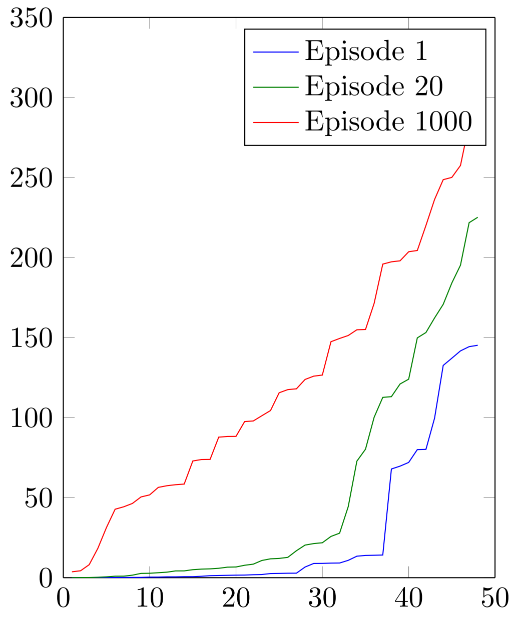


Figure 4.1: Absolute weights for mountain car

From a qualitative viewpoint, it appears that all of the relevance and error measures are well-correlated with the weight. Of interest are the relevance measures: while they do not decrease to zero (due perhaps to the exploration parameter being too low for them to be accurate measurements of the inner products), they still decrease slightly from episode 20 to episode 1000, and show far less structure, while the differences between them and the error measures increase. Both the relevance, and the difference between the observed error and relevance appear to track well with the weights for episodes 1 and 20. There appears to be little difference between the decayed and sample-attenuated variants.

4.7 Discussion and Conclusion

For candidate functions that do not yet form part of the basis function set, we will employ the decayed, sample-attenuated relevance as defined in equation 4.4. This relevance provides a number of attractive theoretical guarantees, and has good empirical performance. For functions already within the basis function set, we will use the difference between the decayed, sample-attenuated observed error $O(\phi)$ and the relevance $\rho(\phi)$, as this tracks well with the weight.

These measures are by no means exhaustive, and represent simple methods for computing approximations to various inner products. Online variance estimates would perhaps also provide a way to measure function relevance, but our focus is on inner product-based measures, rather than statistical approaches, in keeping with the function approximation

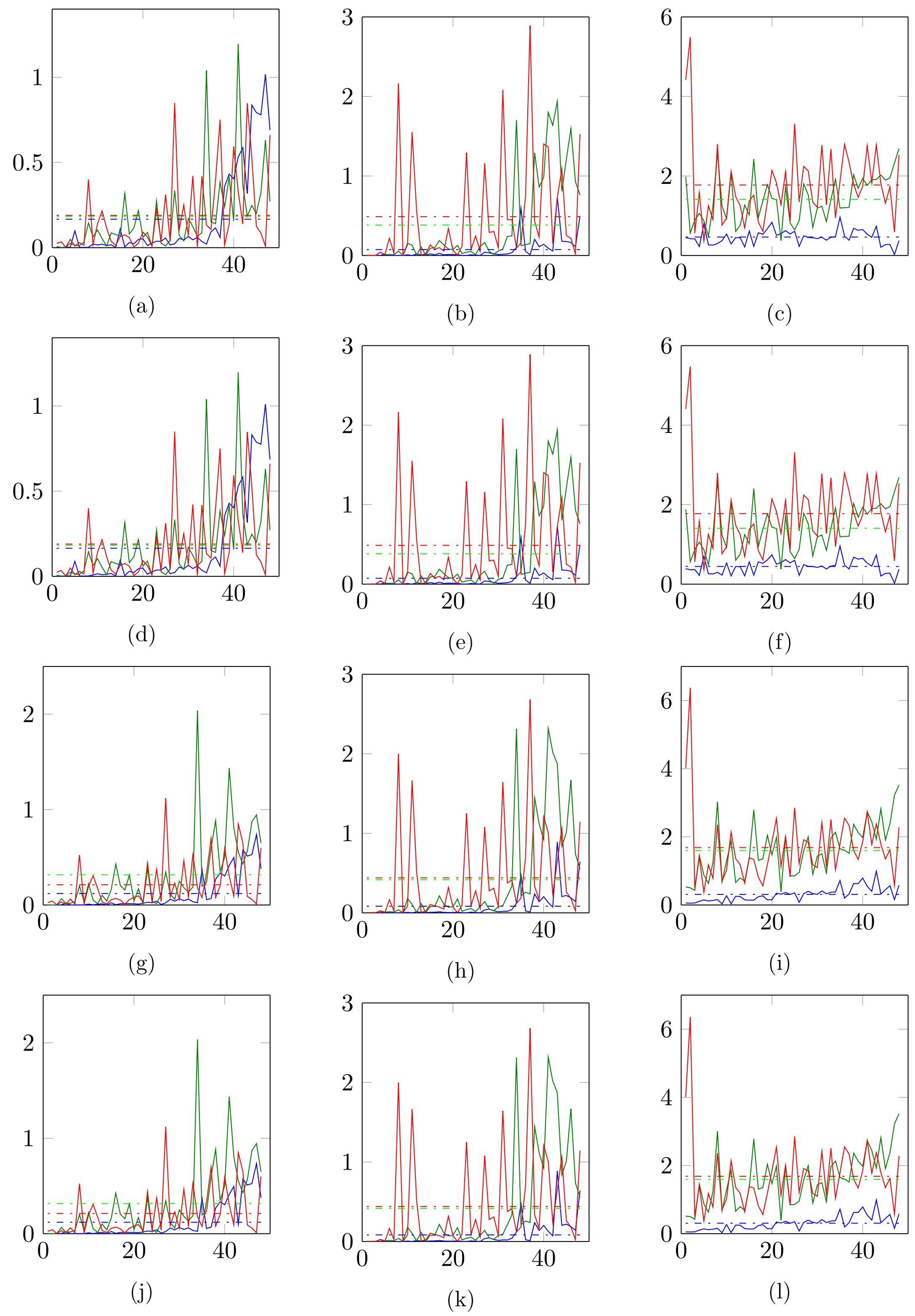


Figure 4.2: Relevances (left), observed error minus relevance (centre) and absolute error minus relevance (right) for measures computed ordinarily (a, b, c), with sample attenuation (d, e, f), with decay (g, h, i) and with both sample attenuation and decay (j, k, l), on the mountain car domain, for 1 episode (blue), 20 episodes (green) and 1000 episodes (red). Average value lines are shown in same colours.

theme of this thesis. In the next chapter, we will use the measures defined here to select candidate functions for detail enhancement.

Chapter 5

Adaptive Wavelet Refinement

5.1 Introduction

The measures of the previous chapter provide us with ways to estimate the relevance of candidate functions in online settings. In this chapter we explore the online Adaptive Tile Coding (ATC) algorithm and similar ideas of adaptive detail refinement within the fields of reinforcement learning, control theory, and numerical analysis, and present an extension of ATC to arbitrary basis function types and learning methods, called Adaptive Wavelet Refinement (AWR), through the use of relevance and error measures.

We prove convergence results, and show that wavelets are both necessary and sufficient for certain classes of value function refinement. We compare our extension to an equivalent fixed basis for both a B-spline basis and a tile coding, across a number of benchmark domains.

5.2 Related Work

Adaptive Tile Coding (ATC) (given in Algorithm 1) [Whiteson *et al.* 2007] is a method for adding representational detail to a tile coding by learning weights for potential tile splits and selecting those splits that maximise the change in either the value function or the policy. It uses TD(0) to learn the weights of both basis functions and potential splits.

Algorithm 1 ATC

```
Require: Tile function set F, state $s$ giving Bellman error $\delta_t$, and splitting tolerance $\tau_s$.
  Update TD(0) weights of all tiles and candidate splits
  if $p$ updates have passed without any $\phi$ encountering its lowest Bellman error $\delta$ then
    for all functions $\phi$ activated by state $(s, a)$ do
      if splitting $\phi$ along $d$ would result in the maximum change across all tiles in either
      the value or policy criterion then
        Replace tile $\phi$ with its child tiles split in dimension $d$ in F.
      end if
    end for
  end if
  return F
```

The value criterion examines the absolute difference between candidate tile weights. If the value function changes rapidly across ϕ in dimension d, one would expect the two child tiles of ϕ split along d to have the largest absolute difference in weight. The policy criterion estimates which tile split would result in a maximal improvement in the policy by counting how many of the $|\mathcal{A}|$ successor states of s would result in a change in policy if split. That is, the algorithm computes $\Delta V_d(s') = w_d(s') - V(s')$, and if altering $V_d(s)$ by $\Delta V_d(s')$ would result in a change in the policy at s, then the count is increased for a potential split in the tile across s' in dimension d. The tile with the largest count is then split along dimension d.

ATC does not offer any theoretical guarantees, but performs well empirically in puddle world and mountain car. If one started with a single tile spanning all dimensions, ATC would scale linearly with the dimensions. If one starts with any other form of fixed basis, ATC would start with a basis function set exponential in dimension. The number of weights to track (including those of the basis function set) is always exactly $|F|(2d + 1)$.

ATC uses TD(0) to compute the function weights. This makes it easy to update the child tile weights, since these must be updated under the assumed absence of their parent tile (or sibling tiles), as splits are performed through replacement of the parent, and preclude splitting that tile in any other dimension. If the parent is updated through

$$\Delta w = \alpha \max_a (R(s,a) - \gamma V(\mathcal{T}(s,a))) - V(s),$$

then the child is updated through

$$\Delta w_d = \alpha \max_a (R(s,a) - \gamma V(\mathcal{T}(s,a))) - w_d.$$

Note the use of w_d in place of $V(s)$: this additionally implies that the tiles are of unit value across their domain. Learning child tile weights using Sarsa(λ) or similar would be more complicated, as the change in weight and trace must be computed without influence from the parent function. Furthermore, one is constrained to using binary functions in the above formula, and the convergence heuristic, value function criterion, and policy criterion do not have a solid theoretical foundation.

Evolutionary Tile Coding (EvoTC) [Lin and Wright 2010] extends this work by allowing the tile split to take place at any arbitrary position. The optimal tiling is computed using a genetic algorithm (GA). The GA takes as its initial population a tree representation of a tiling, wherein each leaf node is the Q value of the tile, and each internal node is the position and dimension along which the tile is split. The fitness of a tiling is evaluated using the overall performance of a learning method using the tiling as a basis function set. Unlike the ATC method, EvoTC has a convergence guarantee, as the GA will (in the limit) explore the space of all possible tilings.

Ideas of adaptive discretisation have had a longer history in control theory and numerical analysis. Munos and Moore [2000] and later Munos and Moore [2002] introduced a variable resolution technique based on value and policy criteria similar to those of ATC. The discretisation procedure explicitly starts with a single tile covering the whole state space, although within each tile Kuhn interpolation is used to achieve linear interpolation (thus, the represented value function is piecewise linear, rather than piecewise constant). The cost of this is that each tile is composed of $d!$ simplexes, leading to factorial complexity with dimension.

Adaptive mesh refinement (AMR), or adaptive gridding or discretisation, appears to have originated in a paper by Melosh and Killian [1976] wherein it was applied to finite element analysis. Since then, it has been applied innumerable times in numerical analysis, particularly when the system may contain singularities, shocks or discontinuities [Berger and Colella 1989] or where much of the system's behaviour is simple, such as in computational astrophysics [Hansen *et al.* 2015]. Much like ATC, AMR techniques rely on *a posteriori* estimates of the approximation error to refine the grid. The interested reader may refer to any text on the subject for further information, as a survey of AMR and adaptive discretisation techniques outside of reinforcement learning is beyond the scope of this thesis.

5.3 Adaptive Wavelet Refinement

Whiteson *et al.* [2007] presented ATC within the context of a TD-0 learning algorithm using tile coding. We extend this algorithm as follows: we allow for any refineable basis function type (we define later what is meant by refineable), we apply the algorithm to Sarsa instead of TD(0) (although the new algorithm could be used with any learning mechanism), and replace learning the subweights of each candidate tile split with our relevance measure from Chapter 4. The new algorithm is called Adaptive Wavelet Refinement (AWR).

At each timestep, the algorithm updates the observed errors and relevances of every function that is nonzero in the current state-action pair, as well as the relevances of each of those function's potential split child functions. The algorithm then selects the function with the largest difference between the observed error and relevance for splitting (provided this difference is larger than the tolerance), and selects the dimension to split in by choosing the dimension with the largest average relevance among the child functions.

Splits are performed with replacement, meaning that the parent function is replaced by its split children. The parent weight is distributed among the children so as to leave the value function unchanged, and the trace of each child function is initialised to zero. The complete algorithm is presented in Algorithm 2. For the purposes of computing the relevance, the functions concerned are assumed to be normal (that is, square-integrable to one), but this requirement need not carry over to the value function itself.

Algorithm 2 AWR

```
Require: Basis function set $\mathbf{F}$, state $s$ giving TD error $\delta_t$, relevance $\rho$, sample count $T$,
  observed error $O$, and splitting tolerance $\tau_s$.
  for all Functions $\phi$ activated by state $(s, a)$ do
    Update relevance $\rho$, observed error $O$ and sample count $T$ of $\phi$
    for all Candidate children $\phi^d_{j,k}$ of function $\phi$ split in dimension $d$ do
      Update relevance and sample count of $\phi^d_{f,k}$
    end for
  end for
  if $O(\phi) - \rho(\phi) \geq \tau_s$ and $\max_\phi O(\phi) - \rho(\phi)$ then
    Replace $\phi$ with its children $\phi^d_{f,k}$ in dimension $d$ such that $\max_d \frac{1}{|k|} \sum_k \phi^d_{f,k}$, redis-
    tributing the weight associated with $\phi$ among its children.
  end if
  return $\mathbf{F}$, $\rho$, $O$, $T$
```

The requirement that the basis function ϕ is refineable means that it can be replaced with smaller copies of itself at regular intervals. This is expressed as follows:

$$\phi(x) = \sum_k w_k \phi(mx - k), \tag{5.1}$$

where $m > 1$, $k \in \mathbb{Z}$ (only refineable functions can be replaced by child functions without altering the value function). Tiles, polynomials and wavelets are refineable, while RBFs and Fourier terms are not. Note that equation 5.1 means that the function is replaced by uniform copies of itself at regular intervals (as used by ATC), rather than arbitrary splits as in the work of Lin and Wright [2010] or overlaid sets of basis functions without replacement.

5.3.1 Theoretical Results

We now prove that wavelets are both necessary and sufficient for certain kinds of adaptive value function schemes, and will always result in a good basis for ergodic processes.

Theorem 5.1. *If one wishes to split a basis function ϕ into a finite number of smaller copies of itself such that the value function remains unchanged, using equation 5.1 where $m > 1$ and $k \in \mathbb{Z}$, with a finite number of w_k nonzero, it is sufficient to use a compactly supported father wavelet as ϕ.*

Proof. The dilation equation 5.1 is obeyed by all father wavelet functions with refinement mask m [Daubechies 1992]. If a father wavelet has compact support, it will have a finite dilation series. $\square$

Theorem 5.2. *If, further, we require that $\phi(x) \in \mathcal{L}_2(\mathbb{R})$ (that is, the basis function is finite square-integrable), then the use of a father wavelet function is necessary to meet equation 5.1.*

Proof. The above conditions define a father wavelet function [Strang 1989]; any function that meets them has an associated mother wavelet and multiresolution analysis. $\square$

Note that a number of functions are refineable, and thus obey equation 5.1 (without the added conditions), for example all polynomial functions are refineable, but are not in $\mathcal{L}_2(\mathbb{R})$. When considered across the unit interval, polynomial functions are in $\mathcal{L}_2(\mathbb{R})$, but are not generally refineable; for example, the function

$$f(x) = \begin{cases} x & 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

cannot be represented as a sum of functions $f(2x - k)$. The most important property of wavelets within the context of this chapter is that all father wavelets satisfy a refinement (or dilation) equation, wherein a father wavelet at a specific scale can be exactly reconstructed using a finite number of wavelets at the next scale. We now present additional theoretical properties of wavelets and AWR.

Theorem 5.3. *Suppose we are given an episodic MDP with finite reward per time step. Then, for all $\epsilon > 0$ and D-dimensional wavelet bases $\Phi(s)$ forming a frame for $\mathcal{L}_2([0,\,1]^D)$, there exists a scale j and weights w_k such that $||V^*(s) - \sum_k w_k \Phi(2^j s - k)|| \leq \epsilon$, where k is an index across $\mathbb{Z}^D$.*

Proof. Since the described MDP is episodic with finite rewards, the returns must also be finite. This means that $V^* \in \mathcal{L}_2([0,\,1]^D)$, as it is finite. Since the selected wavelet family forms a frame for $\mathcal{L}_2([0,\,1]^D)$, all functions in $\mathcal{L}_2([0,\,1]^D)$ can be arbitrarily well-approximated with a finite number of wavelets [Christensen 2008]. $\square$

Theorem 5.4. *Given an MDP initialised with a complete fixed basis at some detail level, a policy inducing ergodic Markov chains, and a learning method that converges to an optimal weight for any added function ϕ added to the basis, the AWR algorithm will converge to a point where the projected error $\langle|\Delta|, \phi\rangle$ is below the splitting tolerance τ_s for all ϕ. For $\tau_s \to 0$, this implies convergence to $\Delta(s) \to 0$.*

Proof. In the limit as $t \to \infty$, the observed error and relevance of each function in the basis set ϕ will tend to the inner product estimates $\langle|\Delta|, \phi\rangle$ and $\langle\Delta, \phi\rangle$ respectively. As proven in Corollary 4.1, the relevance of a function in the basis function set will tend to zero in the limit, and so the difference $O(\phi) - \rho(\phi)$ will tend to $O(\phi) \to \langle|\Delta|, \phi\rangle$. If this is greater than τ_s, the function will be split, and the process repeated on its children. For $\tau_s \to 0$, functions will be split until $\langle|\Delta|, \phi\rangle$ is zero, implying that $\Delta(s) \to 0$ for all states s, since wavelets can arbitrarily well approximate any function for some scale by Theorem 5.3. $\square$

Note that the proof above assumes the convergence of the underlying learning algorithm for any added ϕ, implying that τ_s is set such that the weights of the basis functions and the error estimates converge sufficiently rapidly compared to the rate of addition of new functions. If one considers an ATC variant which could split all the basis functions on each step, the rate of addition of new functions would be too high for learning to take place. AWR is constrained to splitting at most one basis function per step. Furthermore, upon splitting, the added child functions' observed errors are set to zero, ensuring that a split cannot take place until $O(\phi) > \rho(\phi) + \tau_s$ for ϕ.

5.4 Experimental Results

As ATC is a TD(0) method, we cannot compare Sarsa-based AWR to it directly, for differences in performance could be explained through the different learning methods. Instead, we demonstrate the performance of AWR using Sarsa against two fixed bases. A fixed basis at scale 1 (4 wavelets or tiles per dimension) is used to establish baseline performance, and the average return of AWR at scale 1 is shown against a fixed basis at scale 2 (6 wavelets or tiles per dimension). We wish to demonstrate that AWR achieves the performance of an appropriately scaled fixed basis, even when started with an undetailed basis.

We use Sarsa(λ) with $\lambda = 0.9$, $\epsilon = 0.05$ and $\gamma = 1$. PARL2 α scaling is used [Dabney 2014], with $\alpha_0 = 1$. Results are averaged across 100 experiments for 20 adaptive episodes followed by 80 nonadaptive episodes, in order to demonstrate the short-term adaptation (tying into the idea of early generalisation followed by rapid representation), followed by the long-term performance of the learned basis. All results show the average return measured at each episode, and the average number of basis functions in the basis function set at each episode. Standard error was found to be extremely low due to the large number of experiments run, and is thus left out of all results. The tolerance τ_s was selected via empirical experimentation in each case to provide *reasonable* numbers of additional functions, although we acknowledge that ideally the algorithm would not require such parameter tuning. We leave creation of a parameter-free algorithm to future work.

5.4.1 Discontinuous Room

In the discontinuous room domain, we see a large improvement in performance when AWR is employed for both B-splines and tiles (figure 5.1) although the number of functions added by AWR exceeds the number of basis functions at the next scale (figure 5.2). The decreasing performance of the scale 1 fixed bases may indicate an inability to represent the gap through which the agent must pass to get into the next room.

5.4.2 Mountain Car

Figure 5.3 shows that while the AWR B-spline method does learn as fast as the scale 1 B-spline fixed basis, and achieve results comparable to those of the scale 2 fixed basis, there are instabilities when applied to the mountain car domain. We do not know what caused the sharp drop in average performance on episode 30, or the smaller dips around episode 60. Tiling-based methods perform well in this domain, achieving the early learning performance of the scale 1 fixed basis, and long-term performance greater than the scale 2 fixed basis. Figure 5.4 shows the number of basis functions used at each episode.

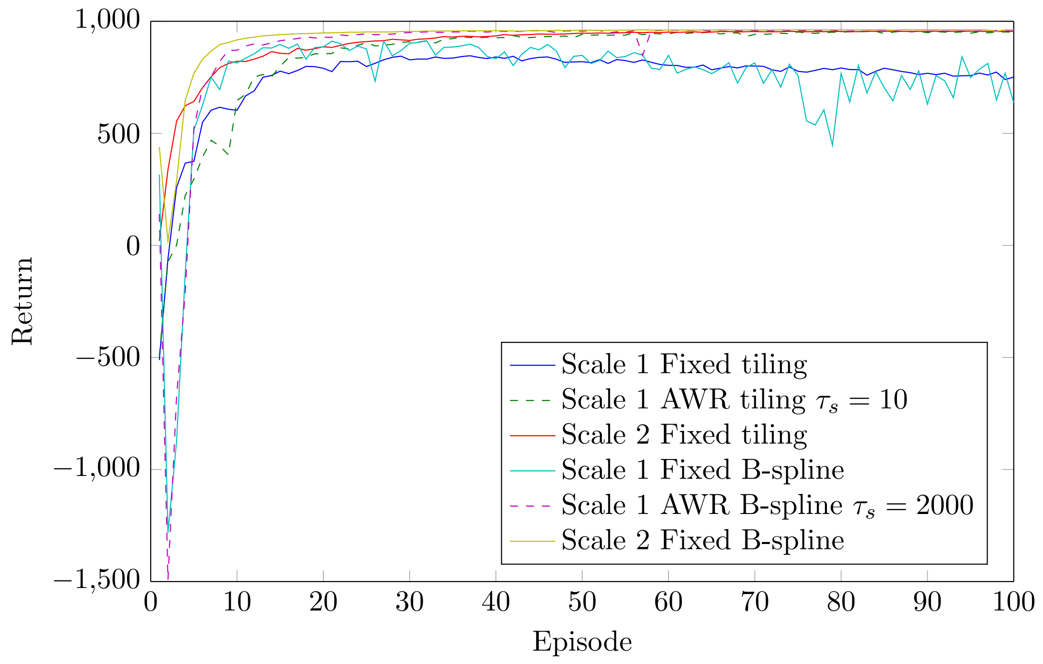


Figure 5.1: AWR Returns for Discontinuous Room

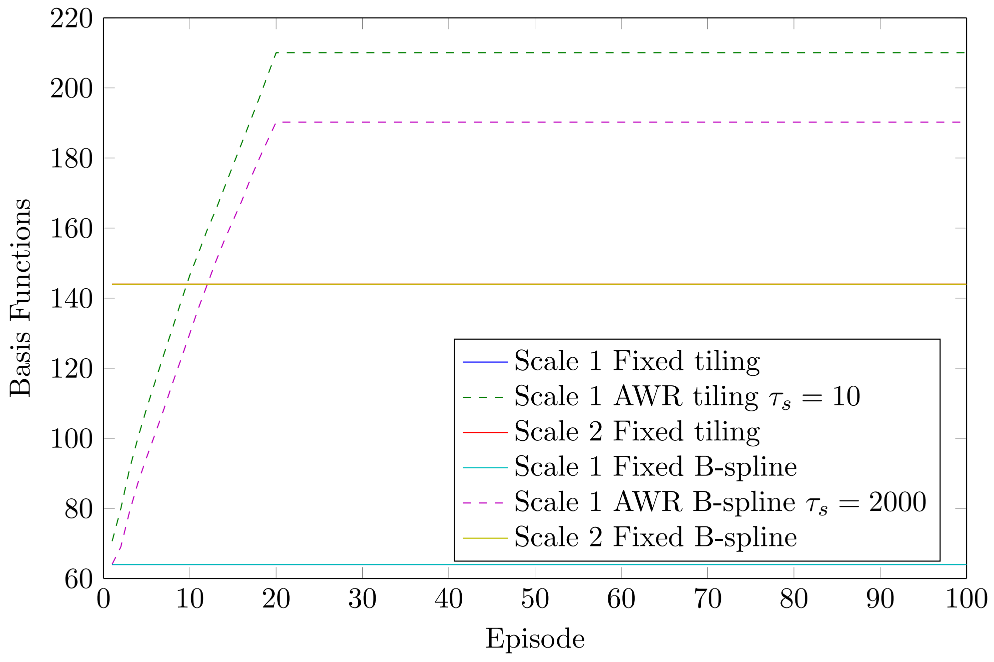


Figure 5.2: AWR Number of Basis Functions for Discontinuous Room

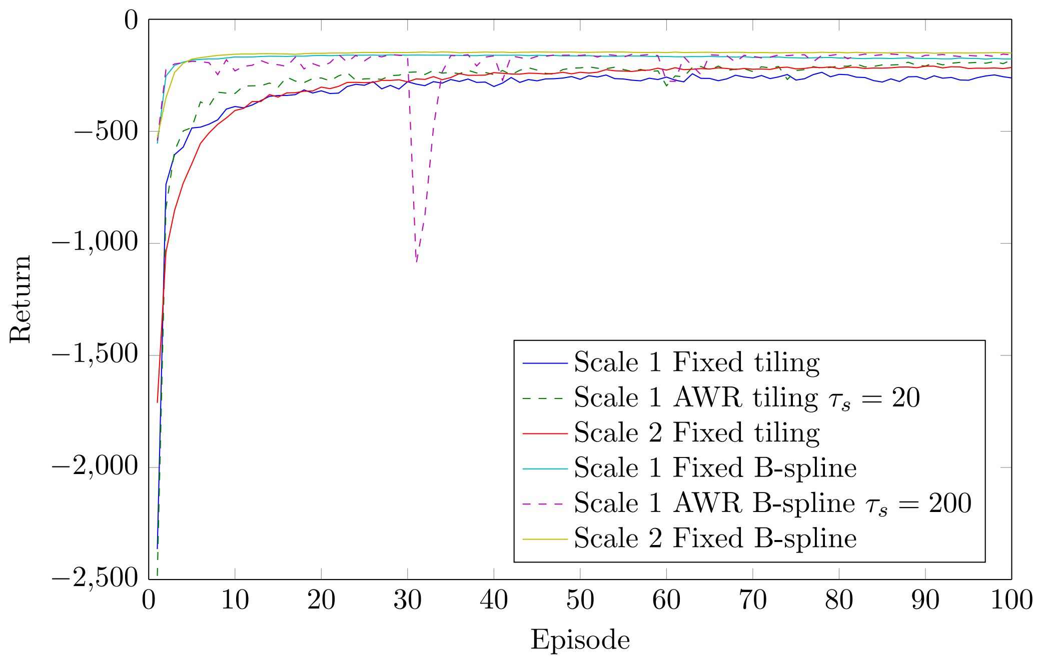


Figure 5.3: AWR Returns for Mountain Car

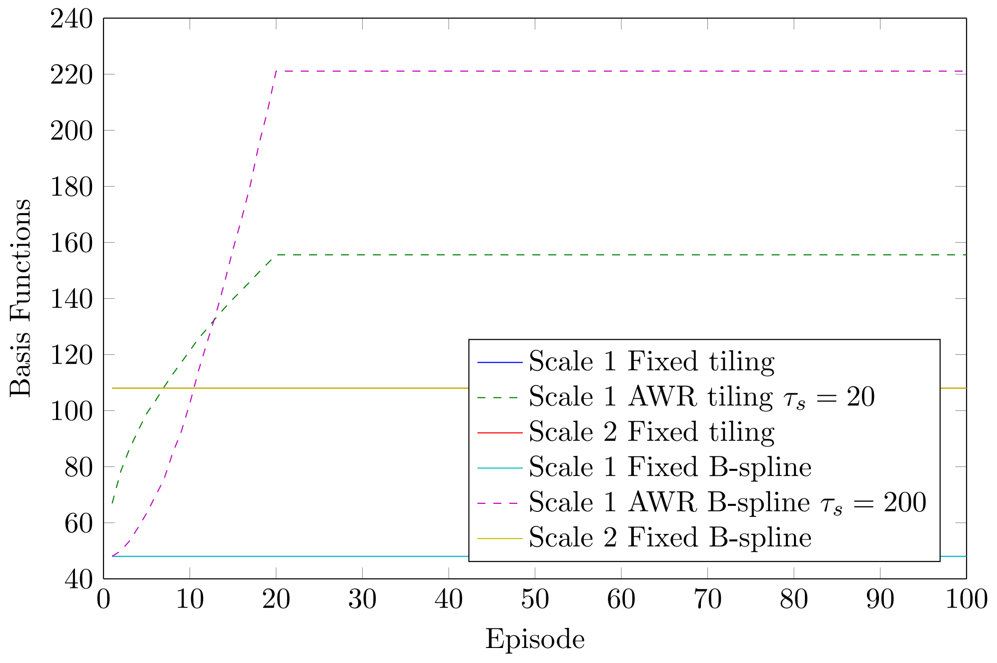


Figure 5.4: AWR Number of Basis Functions for Mountain Car

5.4.3 Acrobot

In figures 5.5 and 5.6 we see that the use of AWR in the acrobot domain did not result in an improvement to the average return. B-spline AWR fails to outperform fixed basis at the same scale: their results are comparable, but the fixed basis performs slightly better overall. The tile-based AWR performs far worse than the fixed basis, indicating an instability of some type.

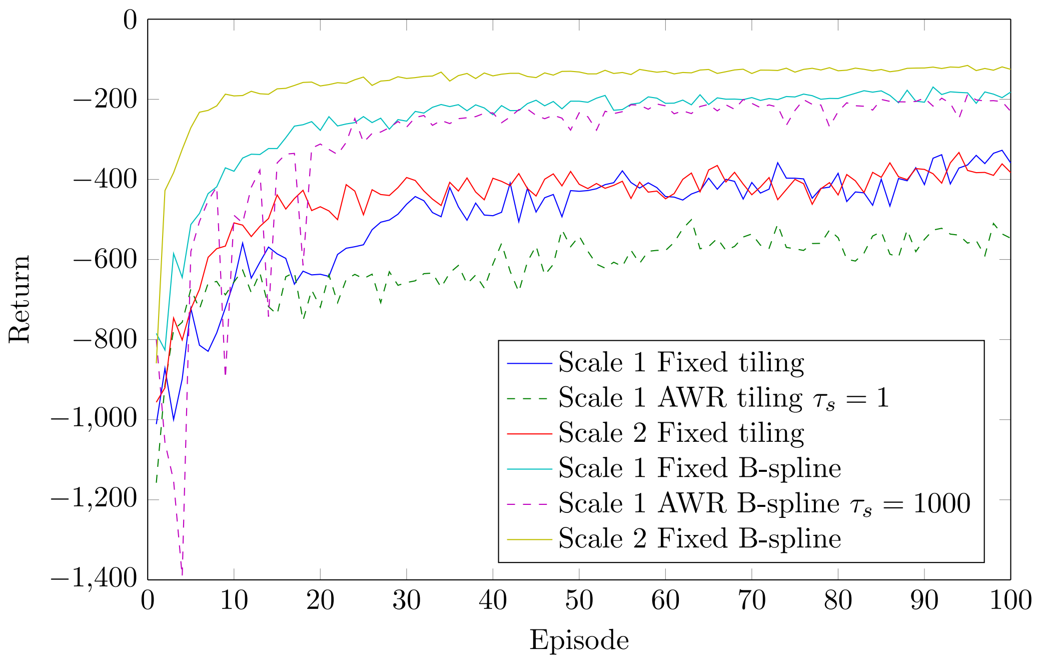


Figure 5.5: AWR Returns for Acrobot

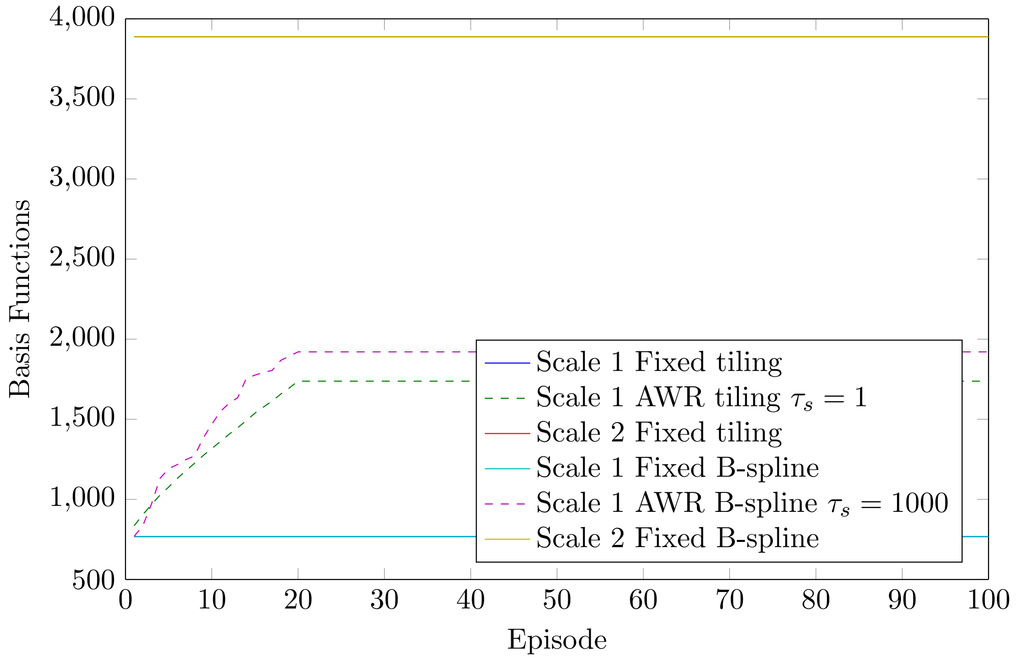


Figure 5.6: AWR Number of Basis Functions for Acrobot

5.4.4 Pinball

In figure 5.7 we see that the AWR method has performed well for both basis function types in the pinball domain. Tiling outperforms the B-spline basis in this domain. Of interest is that in figure 5.8 we see that this increase in performance does not come at the cost of more basis functions than the next scale.

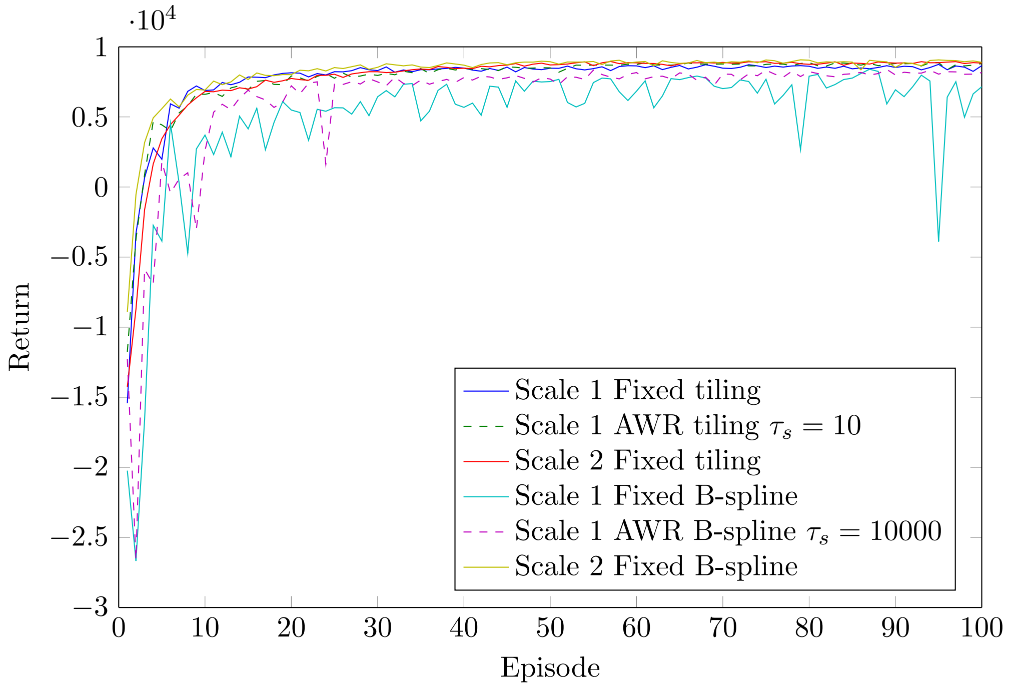


Figure 5.7: AWR Returns for Pinball

5.4.5 3D Mountain Car

Figures 5.9 and 5.10 show a number of strange behaviours in the 3D mountain car domain: the tile-based methods did not converge for either fixed or adaptive methods, the scale 1 fixed B-spline basis outperformed the scale 2 B-spline basis, and the AWR B-spline basis performs as well as the scale 1 basis for the adaptive episodes, and then exhibits large, intermittent performance dips for certain non-adaptive episodes. We suspect the scale 1 fixed B-spline basis matches closely the optimal value function of this domain, such that learning is forced in the direction of the optimal value function regardless of exploratory moves. Deviation from this representation decreases performance.

5.5 Discussion

The results of this chapter are summarised in table 5.1.

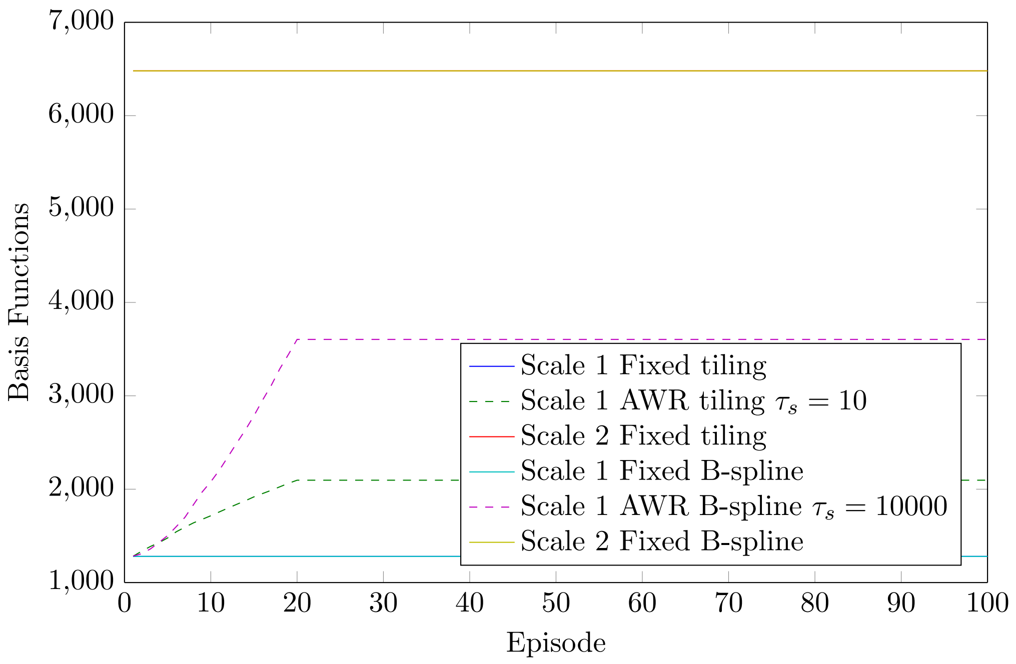


Figure 5.8: AWR Number of Basis Functions for Pinball

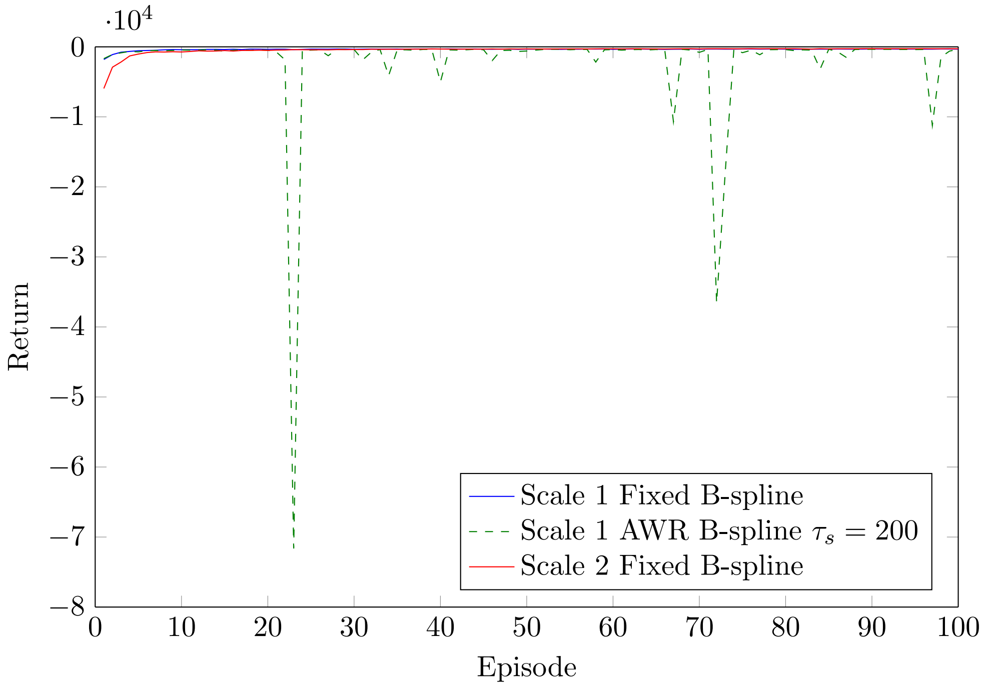


Figure 5.9: AWR Returns for 3D Mountain Car

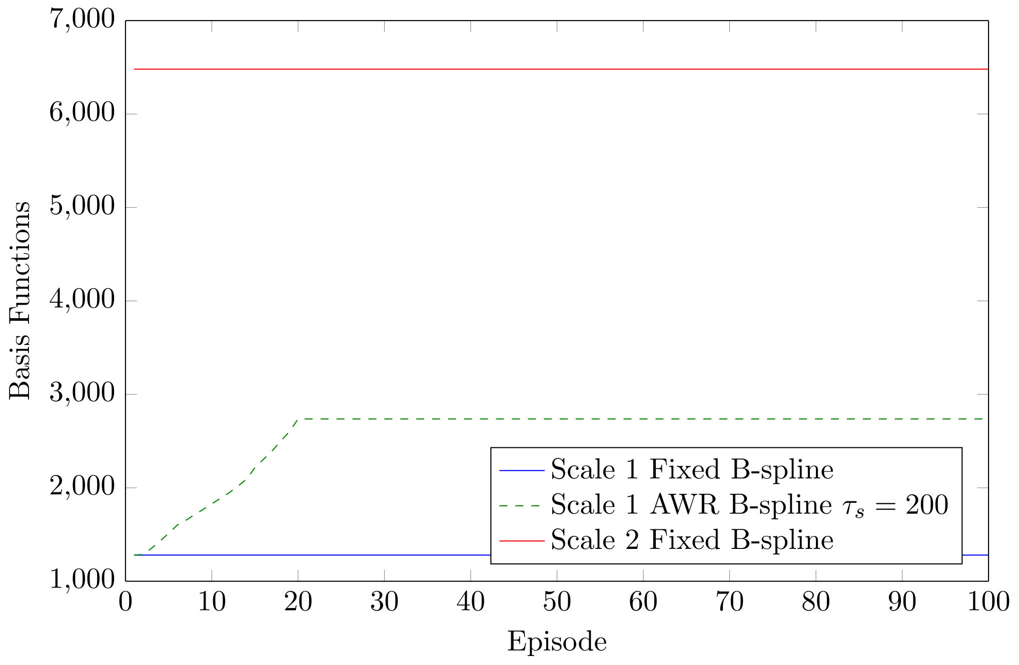


Figure 5.10: AWR Number of Basis Functions for 3D Mountain Car

	AWR vs. Fixed Basis	B-Splines vs. Tile Coding
Discontinuous Room	AWR better	B-Splines better
Mountain Car	AWR better	B-Splines better
Acrobot	AWR worse	B-Splines better
Pinball	AWR better	Tiling better
3D Mountain Car	AWR worse	B-Splines better

Table 5.1: AWR results summary

In acrobot, initial instabilities in the adaptive episodes appear to affect the long-term performance of AWR, dropping it below those of the baseline fixed bases. It is not known what causes this instability, but we conjecture that it is related to inadequate exploration in a small number of experimental runs. In 3D mountain car AWR initially performs well, matching the performance of the scale 1 fixed basis. The instability seen in certain episodes arises in the non-adaptive episodes, and does not appear to affect the overall long-term performance too badly. Nonetheless, for this domain, we cannot say that AWR performs better than the fixed basis. The successes of the AWR method, particularly in the representationally complex pinball and mountain car domains, show the promise of detail-based basis function adaptation.

5.6 Conclusion

We have presented the AWR algorithm, an extension of the method of Whiteson *et al.* [2007] to refineable basis functions with any learning method. This algoritm provides numerous theoretical guarantees, including a convergence theorem conditioned on the convergence of the learning method. Under some assumptions, we have shown that wavelet basis functions are both necessary and sufficient for a broad class of detail-based adaptation with replacement.

AWR has, however, the major disadvantage of starting from a complete fixed basis.

This means that, while the initial detail level may be low, the size of the basis function set will be exponentially dependent on the number of dimensions of the domain. In the following chapter, we will examine dimensional adaptivity to mitigate this problem.

Chapter 6

Incremental Basis Function Dependency Discovery

6.1 Introduction

The method introduced in the previous chapter adds detail incrementally to the basis function set, but cannot scale subexponentially with dimension. In this chapter, we examine an online orthogonal matching pursuit (OMP) algorithm which discovers binary feature dependencies incrementally. This algorithm mitigates the curse of dimensionality by only adding features that are relevant to the problem, using the relevance measure discussed in Chapter 4. We discuss this algorithm and related work, and present a novel extension to this algorithm that allows for arbitrary function types. We prove that the algorithm converges, and demonstrate its empirical performance against the state of the art in a number of domains.

6.2 Related Work

Incremental feature dependency discovery (IFDD) (given in algorithm 3) [Geramifard *et al.* 2011] and IFDD+ (given in algorithm 4) [Geramifard *et al.* 2013a] are two related methods for discovering feature dependencies between binary features online. The set of all pairwise conjunctions of binary features currently within the basis function set is used as the set of candidate functions. A relevance is stored for each candidate function, and updated at each step using the current Bellman error δ. Any conjunction with a relevance greater than a tolerance τ is added to the basis function set, and removed from both its parent functions. That is, if a function $\phi_f = \phi_g \cap \phi_h$ is added to the basis function set, we replace ϕ_g with $\phi_g \leftarrow \phi_g \setminus \phi_g \cap \phi_f$, and similarly modify ϕ_h. This ensures that the basis function ϕ_f cannot be accidently added twice to the basis function set, and leaves no overlap between parent functions and their conjunctive children.

Algorithm 3 IFDD

```
Require: binary basis function set $\mathbf{F}$, state $s$ giving Bellman error $\delta$, relevance $\rho(\phi)$, and
  tolerance $\tau$.
  for all pairwise conjunctions $\phi_f = \phi_g\phi_h$ such that $\phi_g$, $\phi_h \in \mathbf{F}$, and $\phi_f(s) \neq 0$ do
    if $\phi_f \notin \mathbf{F}$ then
      $\rho(\phi_f) \leftarrow \rho(\phi_f) + |\delta|$
    end if
    if $|\rho(\phi_f)| \geq \tau$ then
      $\mathbf{F} \leftarrow \mathbf{F} \cup \{\phi_f\}$
      $\phi_g \leftarrow \phi_g \setminus \phi_g \cap \phi_f$
      $\phi_h \leftarrow \phi_h \setminus \phi_h \cap \phi_f$
    end if
  end for
  return $\mathbf{F}$, $\rho$
```

Algorithm 4 IFDD+

```
Require: binary basis function set $\mathbf{F}$, state $s$ giving Bellman error $\delta$, relevance $\rho(\phi)$,
  sample count $T$, and tolerance $\tau$.
  for all pairwise conjunctions $\phi_f = \phi_g\phi_h$ such that $\phi_g$, $\phi_h \in \mathbf{F}$, and $\phi_f(s) \neq 0$ do
    if $\phi_f \notin \mathbf{F}$ then
      $T \leftarrow T + 1$
      $\rho(\phi_f) \leftarrow \frac{\sqrt{T-1}}{\sqrt{T}}\rho(\phi_f) + \delta$
    end if
    if $|\rho(\phi_f)| \geq \tau$ then
      $\mathbf{F} \leftarrow \mathbf{F} \cup \{\phi_f\}$
      $\phi_g \leftarrow \phi_g \setminus \phi_g \cap \phi_f$
      $\phi_h \leftarrow \phi_h \setminus \phi_h \cap \phi_f$
    end if
  end for
  return $\mathbf{F}$, $\rho$
```

IFDD provides a number of theoretical guarantees, which are reproduced below without their associated proofs. Theorems that only apply to discrete-state MDPs are omitted.

Theorem 6.1. *Given initial features and a fixed policy that turns the underlying MDP into an ergodic Markov chain, TD-based IFDD is guaranteed to discover all possible feature conjunctions or converge to a point where the TD error is identically zero with probability one [Geramifard* et al. *2011].*

Corollary 6.1. *If at each step of IFDD-TD the policy changes but still induces an ergodic Markov chain (e.g., via ϵ-greedy or Boltzmann exploration), then IFDD-TD will explore all reachable features or converge to a point where the TD error is identically zero with probability one [Geramifard* et al. *2011].*

The IFDD and IFDD+ algorithms require binary features. While tile coding produces binary features, other basis function types cannot be used with it directly. From the proof structure followed in Geramifard *et al.* [2011], it would seem that IFDD was originally intended for use in discrete state MDPs. This is supported by observing that the proof of Corollary 3.3 of that paper states that a full expansion in IFDD would result in each state having exactly one active feature, a property only achievable if there are finitely many states.

Temporal difference feature evaluation (TDFE) [Bishop and Miikkulainen 2013] uses a genetic algorithm to search through a space populated by feature subsets. Feature subsets are initialised through random selection without replacement from the set of all available features until the desired subset size is reached (the size is determined by sampling from a normal distribution of a user-supplied mean and variance). This continues until a user-supplied population size is reached. Evaluation then occurs through off-policy value function estimation, and selection is performed to keep the population at a steady size. This algorithm is somewhat similar to IFDD. Both search through the set of all feature subsets, but while IFDD explores a pairwise-conjunctive frontier of features, TDFE explores the set randomly, relying on crossover and mutation. Both work within frameworks that offer convergence guarantees: GA through eventual exploration of the entire space via mutation, MP through convergence in the span of the feature column space.

6.3 Incremental Basis Function Dependency Discovery

The IFDD approach can only be used with binary functions. While one may use tile coding as a binary function set on continuous domains, and in this way mitigate the curse of dimensionality by tiling each dimension independently as the initial feature set, the resulting value functions can only ever be piecewise constant. We now extend IFDD+ to arbitrary function types in algorithm 5. Primarily, we use functions $\phi_i(s)$ in place of binary features, giving the new algorithm the name Incremental Basis Function Dependency Discovery (IBFDD).

Algorithm 5 IBFDD

```
Require: Basis function set $\mathbf{F}$, state $s$ giving TD error $\delta_t$, relevance $\rho(\phi)$, sample count
  $T(\phi)$, tolerance $\tau_c$ and the size of the domain of the function $\phi$ within the area of interest,
  $|\Omega_\phi|$
  for all Pairwise conjunctions $\phi_f = \phi_g\phi_h$ such that $\phi_g$, $\phi_h \in \mathbf{F}$, and $\phi_f(s) \neq 0$ do
    if $\phi_f \notin \mathbf{F}$ then
      Update $\rho(\phi_f)$
      Update $T(\phi_f)$
    end if
  end for
  if $\rho(\phi_f) \geq \tau_c$ and $\max_f \rho(\phi_f)$ then
    $\mathbf{F} \leftarrow \mathbf{F} \cup \{\phi_f\}$
  end if
  return $\mathbf{F}$, $\rho$, $T$
```

The primary differences between IFDD+ and IBFDD are that we add only the conjunction with the highest magnitude relevance greater than τ_c at each step, and the inclusion of the function values in the relevance measure. This relevance measure is an estimate of the projection of the Bellman error onto the function. The conjunctive functions are not removed from their parent functions in our scheme, as this removal has no clear meaning within the context of arbitrary functions ϕ. As is the case with IFDD, the set of candidate functions is of finite size, and so $\mathbf{F}$ is also of finite size.

Theorem 6.2. *Given a changing policy π that, through exploration, induces ergodic Markov chains, and a learning method that converges to an optimal weight for any added function ϕ, the IBFDD method will converge to a point where the projected error $\langle \Delta, \phi \rangle$ is below the combining tolerance τ_c for all existing candidate functions ϕ. For $\tau_c \to 0$, this implies convergence to either $\delta(s) \to 0$, or the addition of all candidate functions to the basis function set.*

Proof. In the limit as $t \to \infty$, the relevance of each candidate function ϕ will tend to the inner product estimate $\langle \Delta, \phi \rangle$. If this is greater than τ_c, the candidate function will be added, and the process repeated on its conjunctions. For $\tau_c \to 0$, candidate functions will be added until no more candidate functions exist, or $\langle \Delta, \phi \rangle$ is zero, implying that $\Delta(s) \to 0$ for all states s. □

Theorem 6.3. *The IBFDD algorithm may be considered an online approximation of an orthogonal matching pursuit algorithm.*

Proof. A similar result was derived for Batch-IFDD [Geramifard *et al.* 2013b], so we give a less rigorous argument for our new relevance measure. Since the relevance measure is a Monte Carlo approximation of the inner product $\langle \Delta, \phi \rangle_P$, it is an estimate of the projection of candidate function ϕ onto the Bellman error surface Δ. Selecting the largest such relevance measure means the algorithm has the same behaviour as a *matching pursuit* method, working with projection estimates computed in finite time. As the weights associated with the basis functions are subsequently shifted by the addition of the candidate function ϕ, the method is an online approximation of an *orthogonal matching pursuit* algorithm. □

In the worst case, every function in the conjunction set is activated by state s, necessitating a ρ update for every candidate function. In the average case, however, we expect that not every function will be activated. If wavelet functions are used, one can quickly index into the set of activated functions without having to check the entire set of conjunctions. Suppose we wish to find which functions are activated in state $s \in [0, 1]^D$, where s_d will denote the state element in dimension d, given a set of conjunctions of univariate wavelet functions $\phi_{j,k} = \phi(2^j x_d - k_d)$. Since the domain of the atomic wavelet function ϕ is known (say, $[0, a]$), we can for a specific j and s_d generate a list of translations k_d in time linear in the order of the wavelet function such that in dimension d, s_d is in the domain of $\phi(2^j x_d - k_d)$. This is possible because the domain of $\phi(2^j x_d - k_d)$ is $[\frac{k_d}{2^j}, \frac{a+k_d}{2^j}]$ and so the k_d such that s_d is within this domain is given by $k_d \in \{\lfloor 2^j s_d \rfloor, \ldots, \lfloor a + 2^j s_d \rfloor\}$.

A smart index is maintained as a list of hash tables, indexing into lists of basis functions. The outermost list gives a hash table for each dimension d, and the hash table indexes translations k_d into lists of basis functions ϕ, such that $\phi(2^j s_d - k_i)$ is an atomic function in each ϕ in the list. An empty list means no function is stored at that location for that dimension. A state s is examined in each dimension, and for each possible translation k_d in that dimension, the corresponding hash table entries at k_d are extracted as potentially activated functions, since these functions must still be checked against the entirety of s. Each conjunction is stored in one hash table only, which in our implementation corresponded to its first active dimension. A more efficient implementation would be to store it in the hash table of a random active dimension, to ensure the hash tables are roughly balanced. The overall complexity of indexing all potentially activated functions is $O(nD)$, where n is the order of the wavelet basis determining the length of the support, a. Since each k may index into a list of conjunctions whose length is in some way dependent on the conjunctive frontier size, an average-case complexity analysis would be difficult. The number of functions checked is guaranteed to be less than or equal to the size of the set of candidate functions, and in practice is much smaller.

6.4 Empirical Results

The IFDD and IFDD+ algorithms are tested against IBFDD employing tile coding, and IBFDD using order 2 B-spline wavelets, at scales 1 and 2 (or equivalently 4 or 6 functions per dimension), in five domains. We use Sarsa(λ) with $\lambda = 0.9$, $\epsilon = 0.05$ and $\gamma = 1$.

PARL2 α scaling is used [Dabney 2014], with $\alpha_0 = 1$. Results are averaged across 100 experiments for 20 adaptive episodes followed by 80 nonadaptive episodes, in order to demonstrate long-term performance of the learned basis. All results show the average return measured at each episode, and the average number of basis functions in the basis function set at each episode.

In the original paper, the IFDD relevance [Geramifard *et al.* 2011] appears to contain an error, in that the relevance is given as the sum of the absolute value of the error, rather than the absolute value of the sum of the error. Since this is corrected in subsequent publications [Geramifard *et al.* 2013ab], we will instead use the corrected version when IFDD is used. The removal of conjunctions from the parent function was implemented as a taboo list when checking the value of the parent tile function. If the child function is activated, the parent function is not. In all domains, IFDD and IFDD+ exhibited a marked sensitivity to the tolerance parameter: either a large number of functions were added, or none were. The tolerances of the extension methods were chosen to attempt to match the number of basis functions selected by the IFDD method.

6.4.1 Discontinuous Room

Tile-based IBFDD and IFDD did not converge for this domain. We show results for B-spline based IBFDD against the tile-based IFDD+ method in figures 6.1 and 6.2. When 4 B-splines per dimension are employed, a stable basis is not achieved (recall the instabilities seen in the fixed basis at scale 1), while 6 B-splines per dimension produce performance surpassing that of the tile coding bases, while adding fewer functions. This indicates the weakness of the IFDD approaches: if the initial level of detail is insufficient, the basis will never be adequate, even if every conjunction is added. As with the fixed basis results, tile coding does not exhibit a dip in the second episode.

6.4.2 Mountain Car

Figures 6.3 and 6.4 show that IBFDD using B-splines outperform the extension using tile coding, as well as IFDD and IFDD+, using fewer basis functions in the process. The extension using tile coding additionally has more stable performance than IFDD, improving on the results when performance in the non-adaptive episodes are considered. These trends are also seen when the methods are started with 6 basis functions per dimension (figures 6.5 and 6.6), although one of the nonadaptive episodes using the B-spline basis appear to have performed poorly on average, perhaps indicating very poor performance in a single episode of a single experiment. However, this has not affected the overall results.

6.4.3 Acrobot

Figures 6.7 and 6.8 show that IBFDD with 4 B-splines per dimension outperforms tile coding-based approaches with a roughly equivalent number of basis functions used, but here we see poor performance from the tile coding-based IBFDD. This is also seen in figures 6.9 and 6.10, where six basis functions per dimension are used.

6.4.4 Pinball

Figures 6.11 and 6.12 shows a failure in the B-spline based IBFDD, wherein severe instabilities are seen in the long-term performance when four basis functions per dimension are used. This is not seen when six basis functions per dimension are used (figures 6.13 and 6.14), although the IBFDD methods do not outperform either IFDD or IFDD+ in this domain.

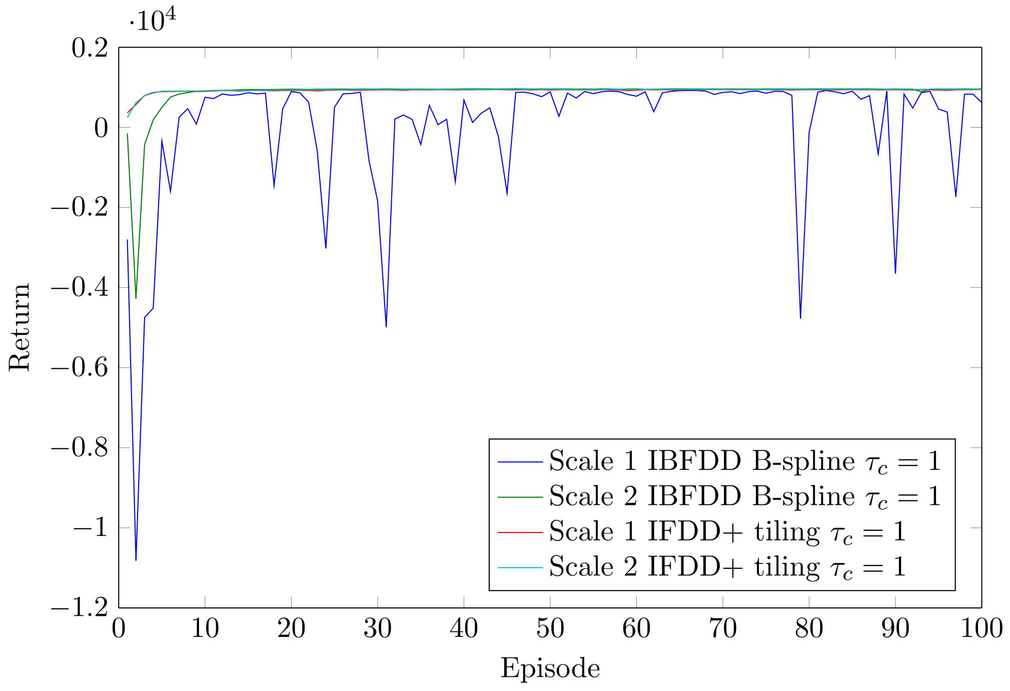


Figure 6.1: IFDD Returns for Discontinuous Room

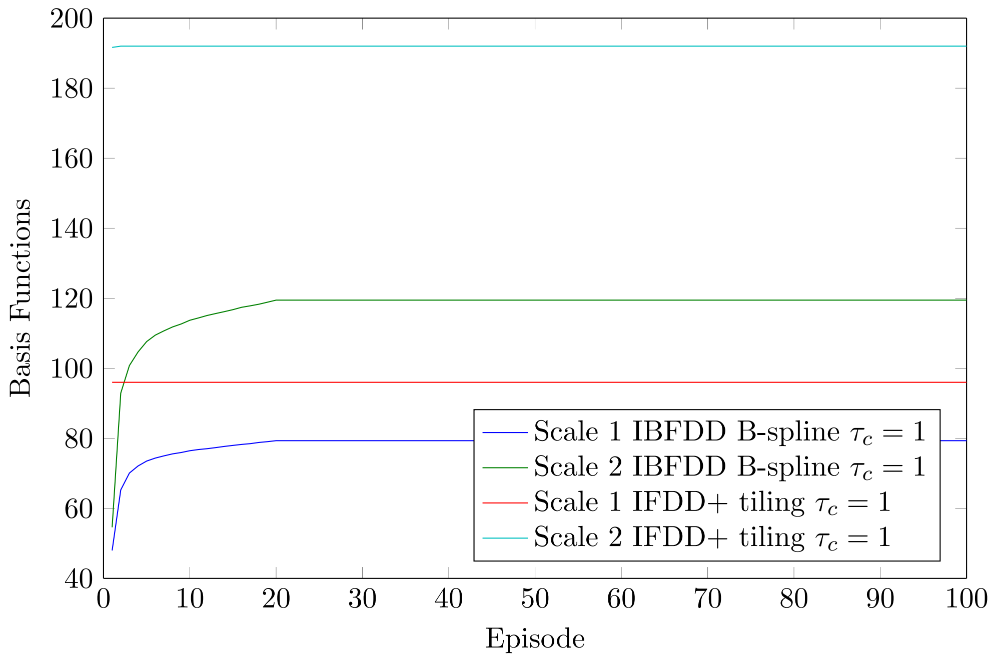


Figure 6.2: IFDD Number of Basis Functions forDiscontinuousRoomIFDD

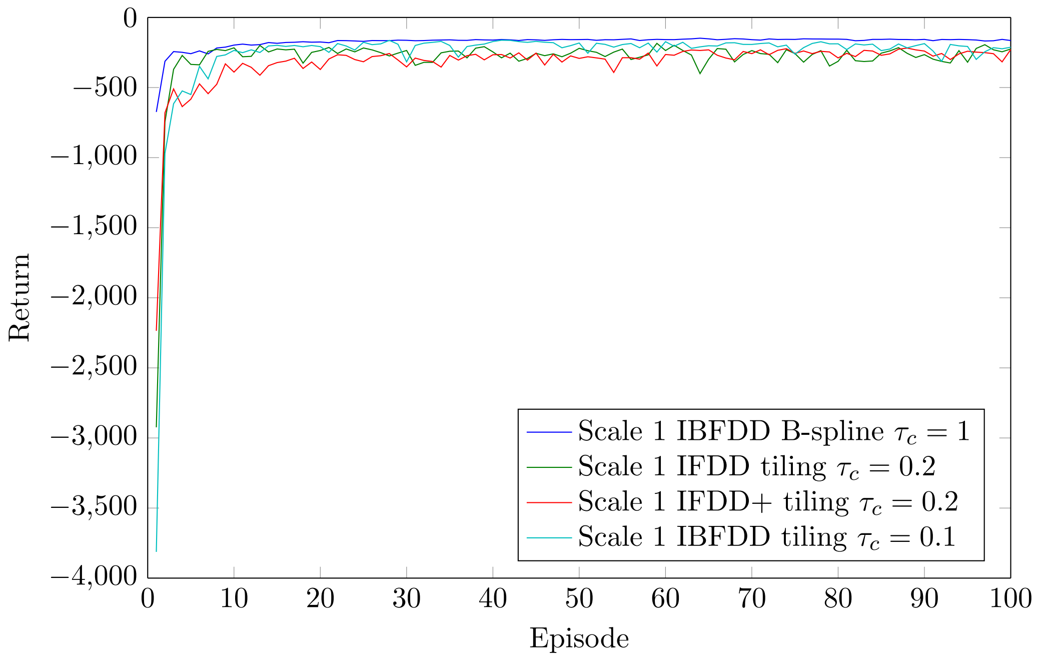


Figure 6.3: IFDD Returns for Mountain Car at Scale 1

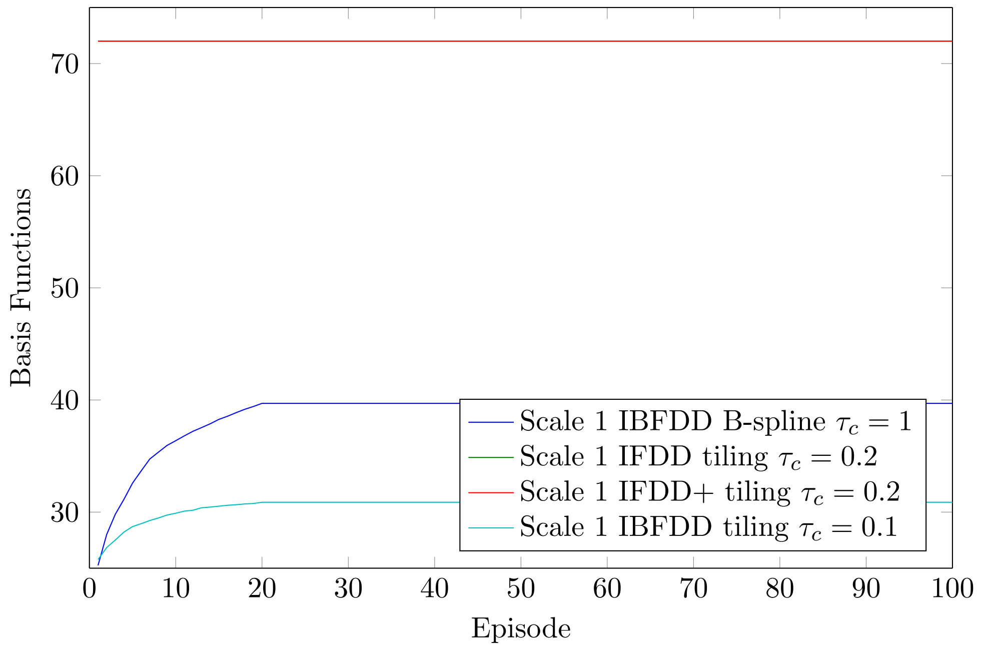


Figure 6.4: IFDD Number of Basis Functions for Mountain Car at Scale 1

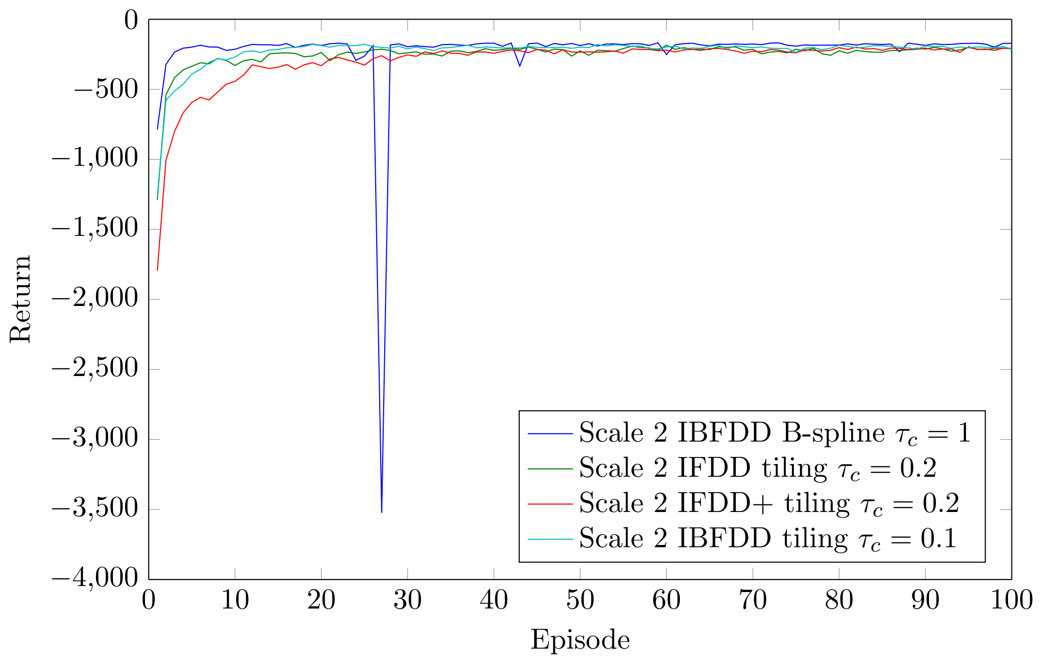


Figure 6.5: IFDD Returns for Mountain Car at Scale 2

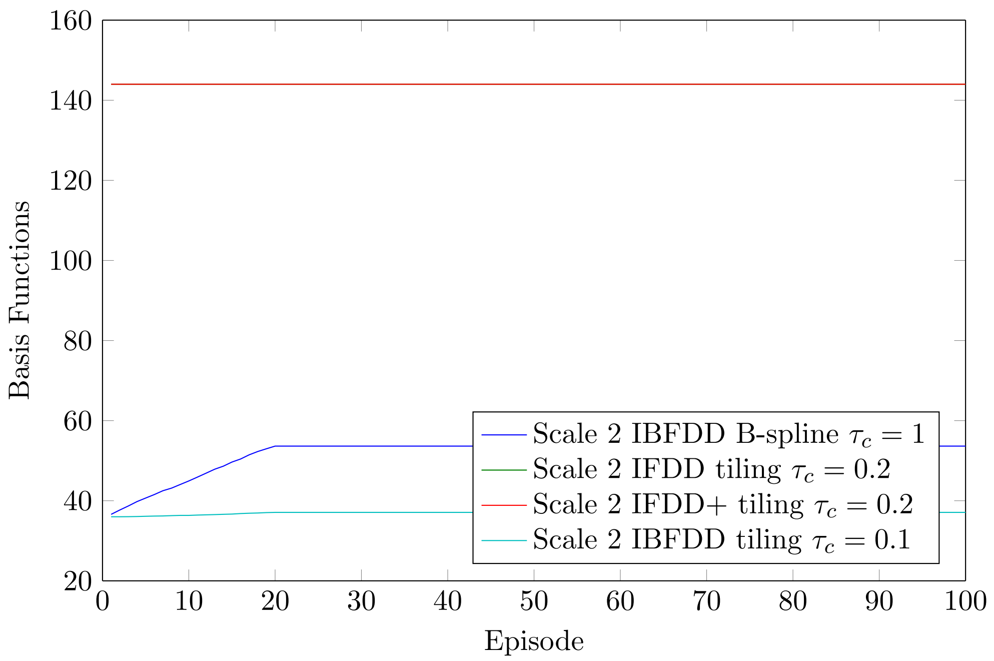


Figure 6.6: IFDD Number of Basis Functions for Mountain Car at Scale 2

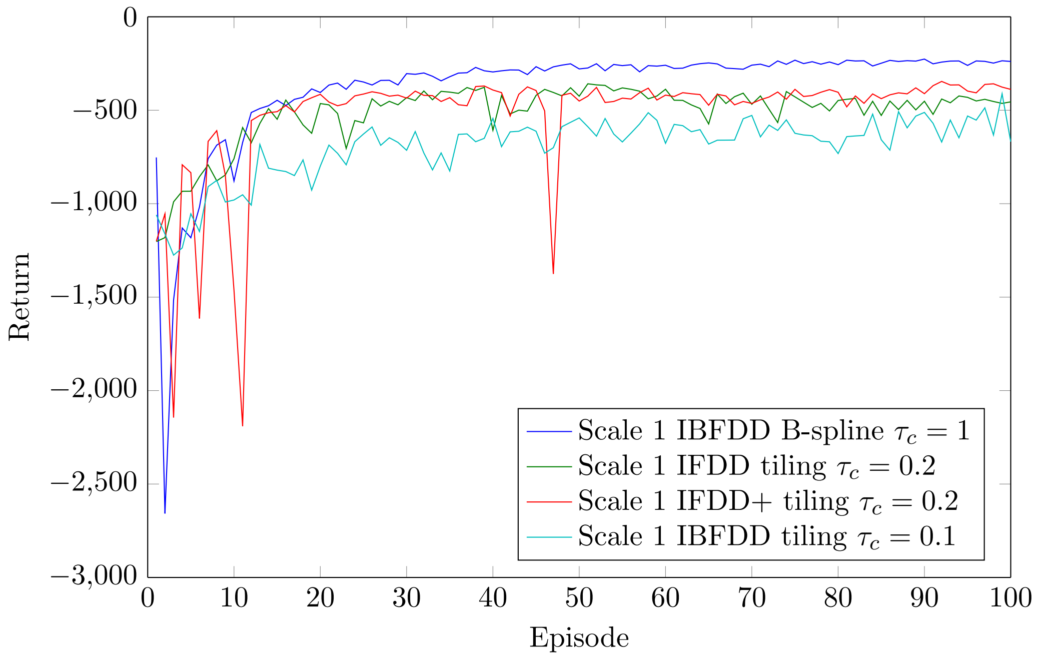


Figure 6.7: IFDD Returns for Acrobot at Scale 1

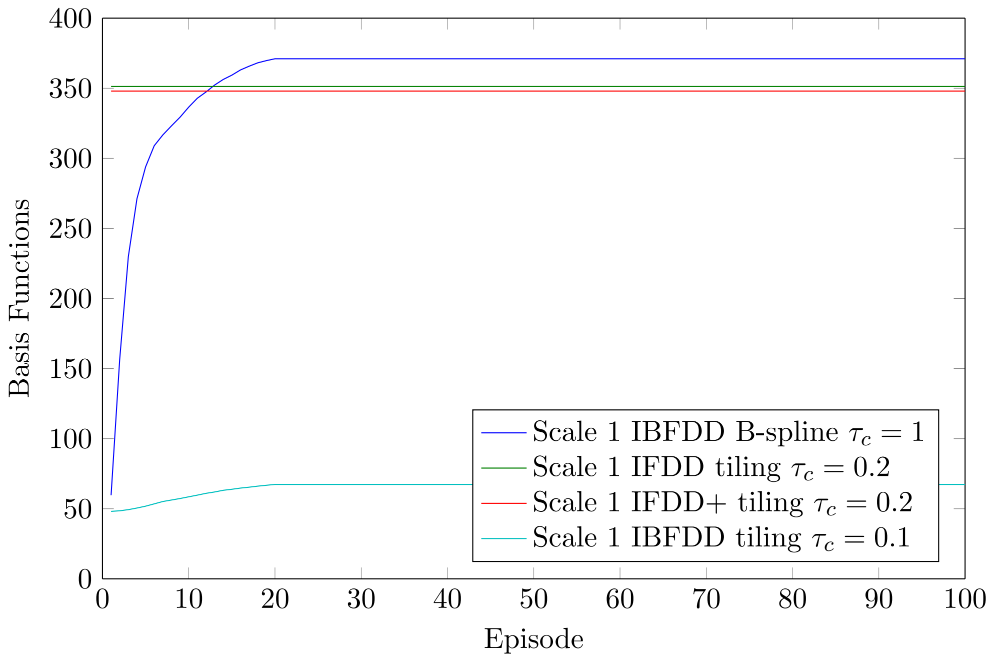


Figure 6.8: IFDD Number of Basis Functions for Acrobot at Scale 1

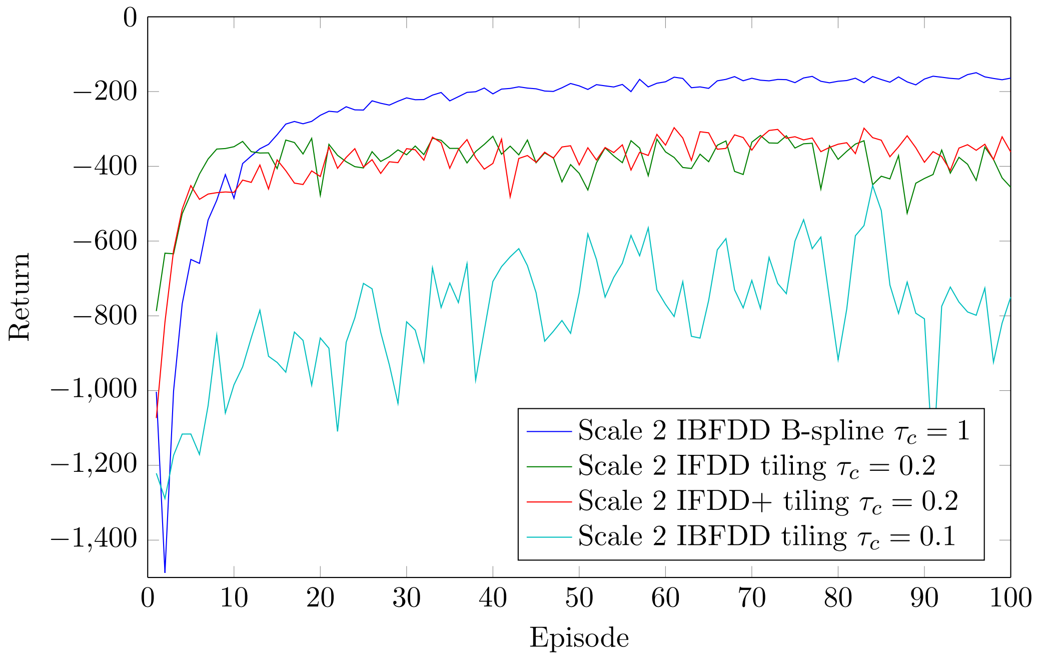


Figure 6.9: IFDD Returns for Acrobot at Scale 2

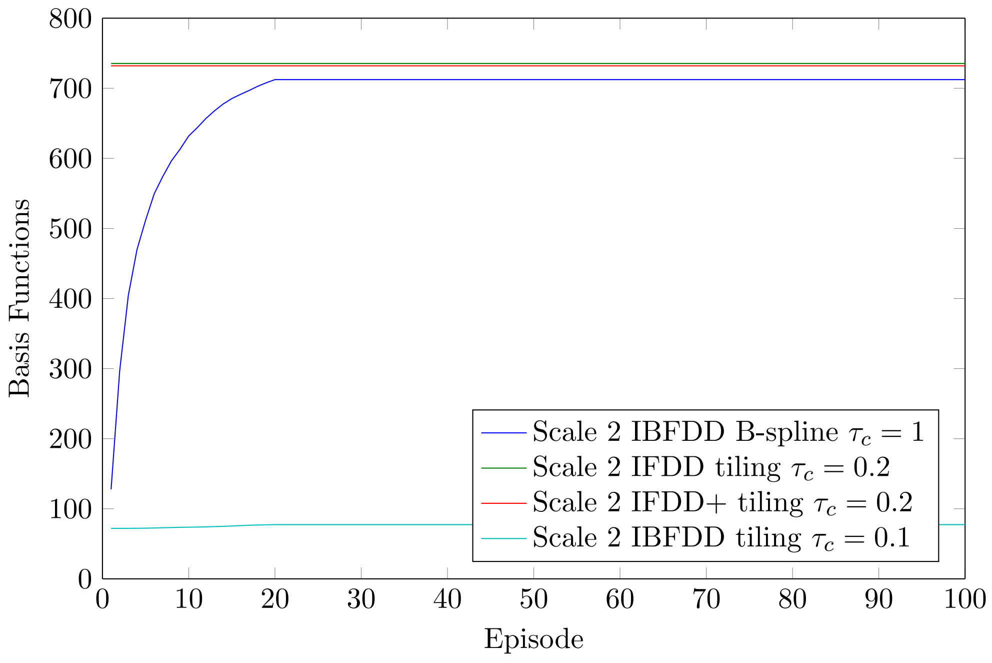


Figure 6.10: IFDD Number of Basis Functions for Acrobot at Scale 2

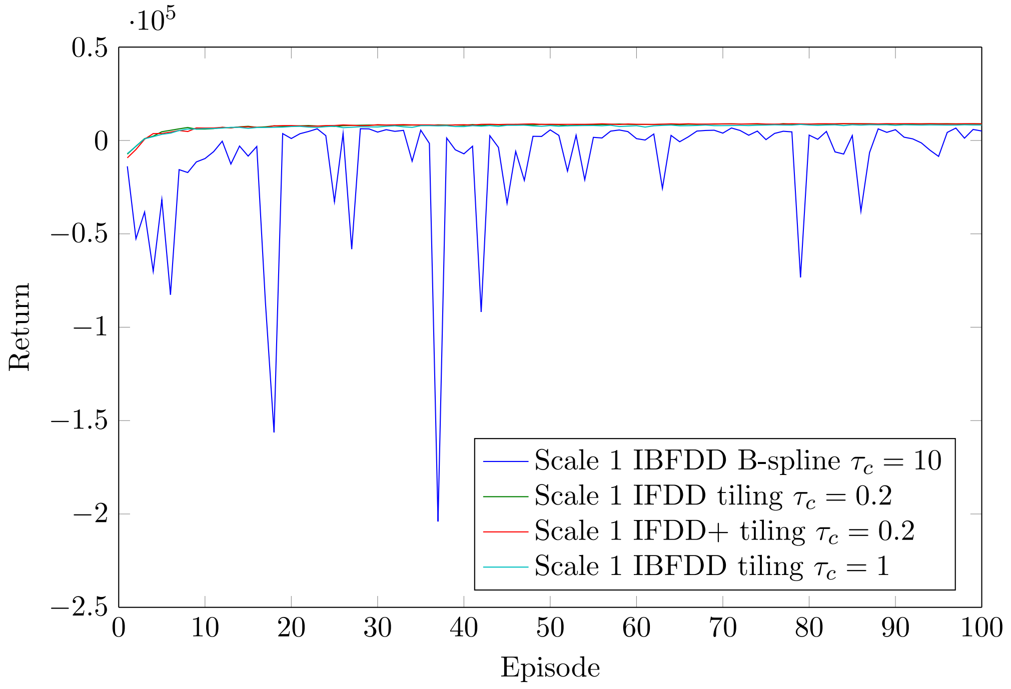


Figure 6.11: IFDD Returns for Pinball at Scale 1

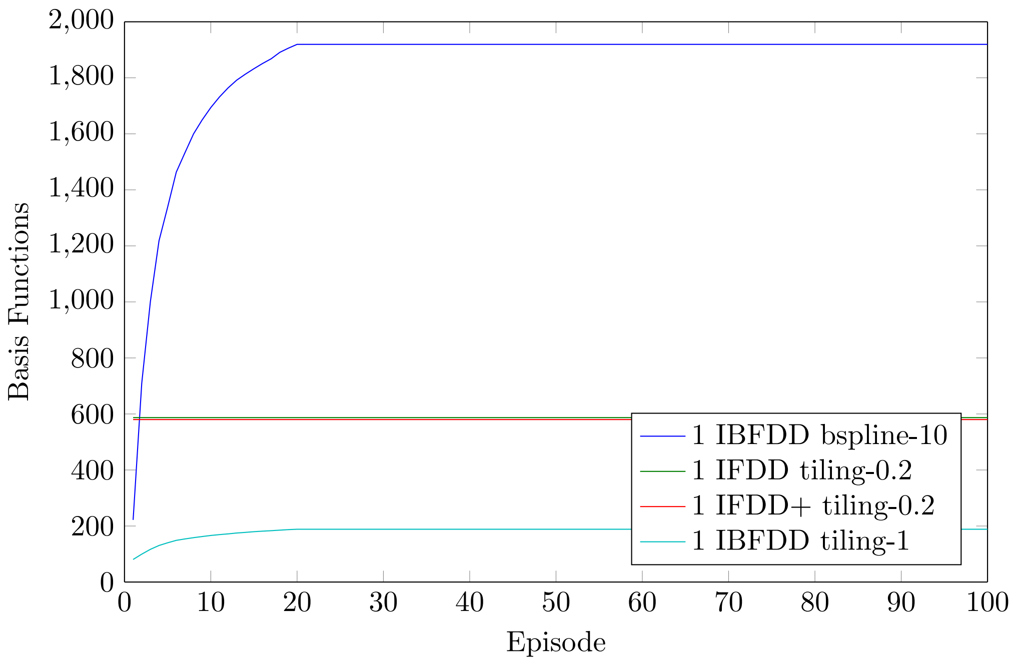


Figure 6.12: IFDD Number of Basis Functions for Pinball at Scale 1

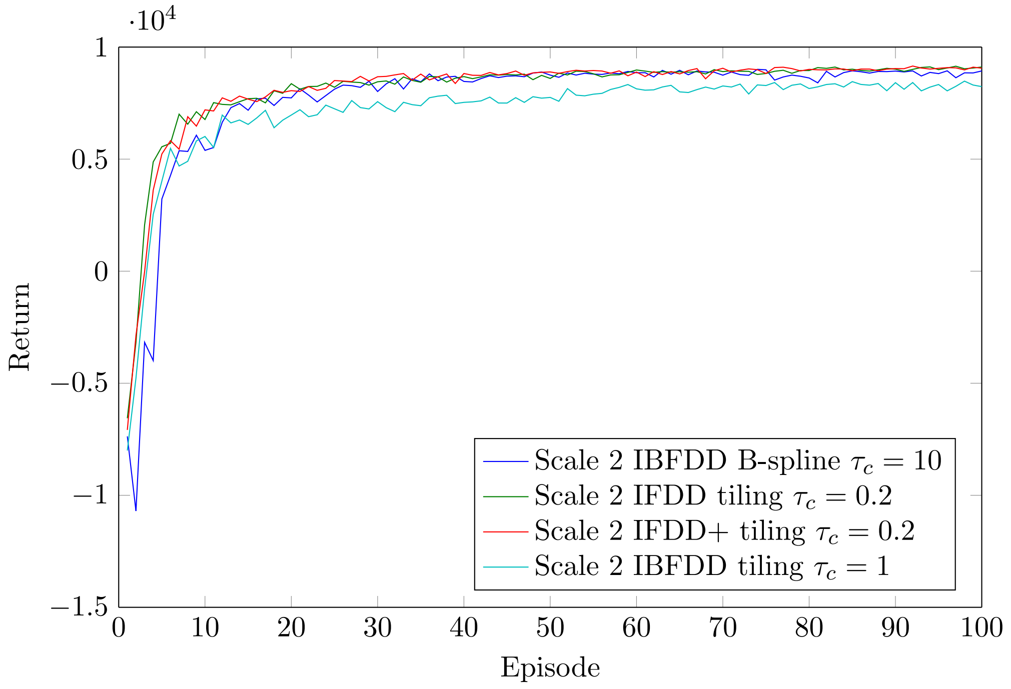


Figure 6.13: IFDD Returns for Pinball at Scale 2

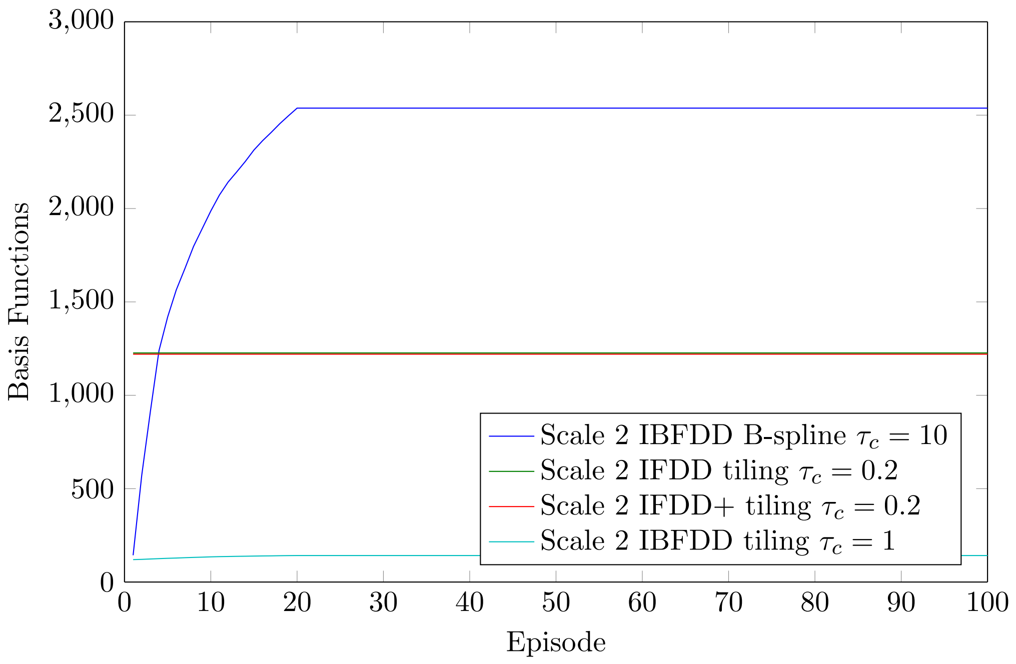


Figure 6.14: IFDD Number of Basis Functions for Pinball at Scale 2

6.4.5 3D Mountain Car

Figures 6.15 and 6.16, and figures 6.17 and 6.18 show that the B-spline IBFDD outperforms the tile-based methods, using a roughly equivalent number of basis functions, for both 4 and 6 basis functions per dimension. We also note that the instabilities seen in the AWR basis are not present here.

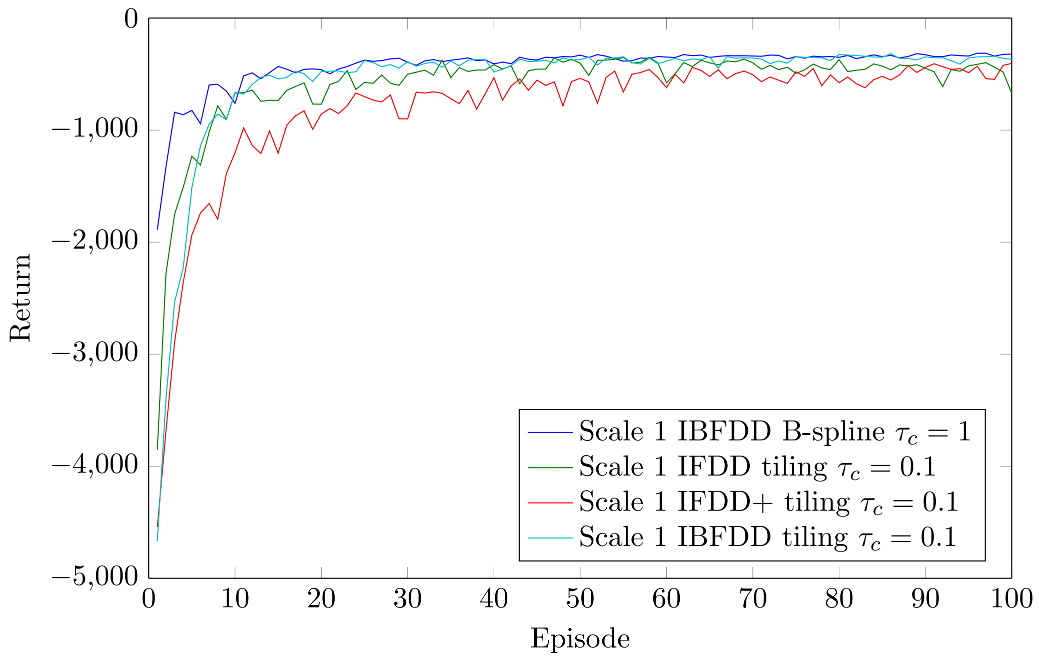


Figure 6.15: IFDD Returns for 3D Mountain Car at Scale 1

6.5 Discussion

Table 6.1 summarises the results above. In three of the five tested domains, the B-spline based IBFDD outperformed the tile-based methods. In the remaining two domains, results were inconclusive due to mixed performance dependant on the starting scale. For scale 2 wavelets (that is, 6 basis functions per dimension), the B-spline based IBFDD outperformed the tile-based methods in every domain except pinball, wherein the performance was roughly equal to the IFDD and IFDD+ methods. The spiking seen in Mountain Car may be attributed to a single episode in a single run performing abysmally, as this does not seem to have unduly affected overall performance.

	IBFDD vs. IFDD/+	B-Splines vs. Tile Coding
Discontinuous Room	Inconclusive	Inconclusive
Mountain Car	IBFDD better	B-Splines better
Acrobot	IBFDD better	B-Splines better
Pinball	IBFDD worse	Inconclusive
3D Mountain Car	IBFDD better	B-Splines better

Table 6.1: IBFDD results summary

We thus conclude that IBFDD with B-splines performs better than IBFDD with tile coding, and overall it seems that IBFDD outperforms IFDD and IFDD+ (although we

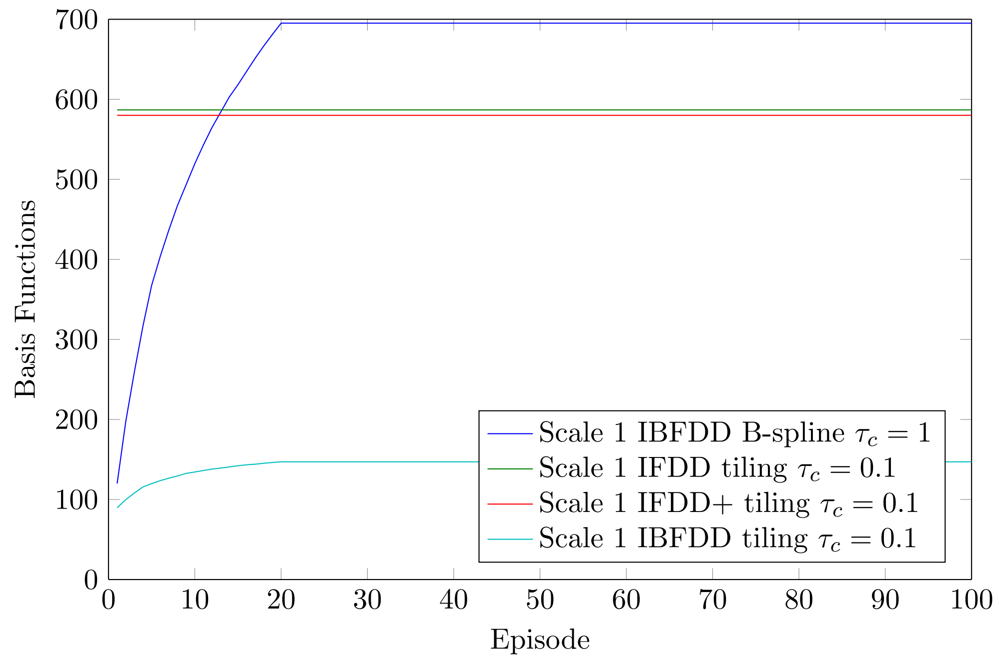


Figure 6.16: IFDD Number of Basis Functions for 3D Mountain Car at Scale 1

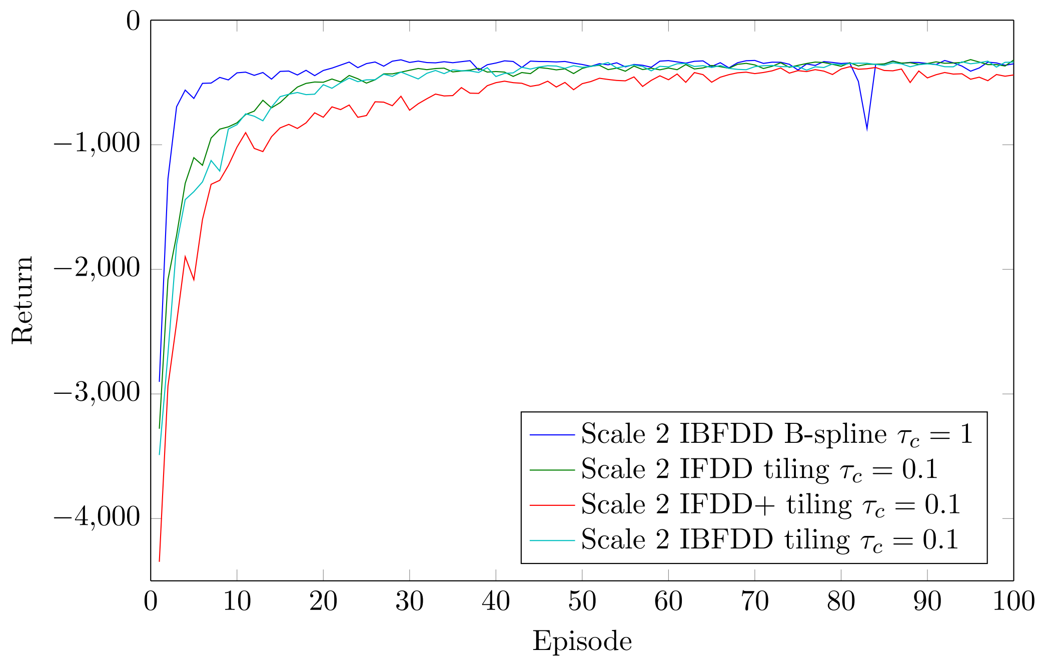


Figure 6.17: IFDD Returns for 3D Mountain Car at Scale 2

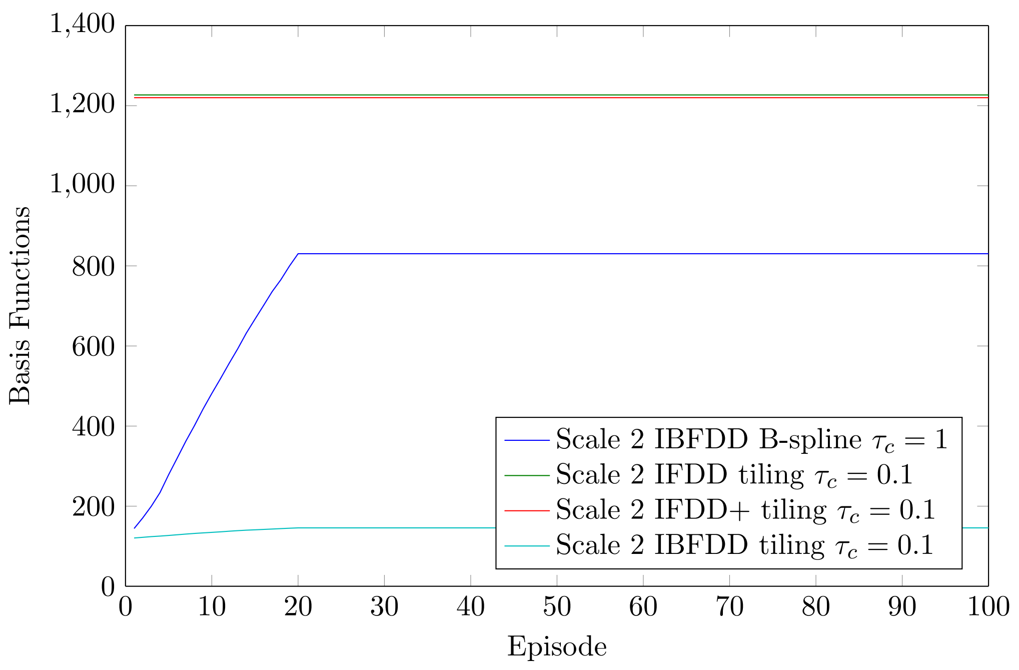


Figure 6.18: IFDD Number of Basis Functions for 3D Mountain Car at Scale 2

do not claim statistical significance of the results). As IBFDD generalises the ideas of IFDD to any basis function set, it has clear advantages over IFDD. Both approaches, however, cannot add representational detail that is lacking from the initial basis function set. As such, we recommend neither method for use within general reinforcement learning problems wherein the appropriate level of detail is not known.

6.6 Conclusion

The IBFDD method (like the IFDD and IFDD+ methods) is dimensionally insensitive, and can be applied to large-scale problems, an advantage over both the fixed basis and the AWR method. However, if the initial level of detail is insufficient, IBFDD may not produce a stable basis, even if the basis is fully expanded. In the following chapter, we present a novel algorithm which combines the dimensional scalability of IBFDD and the detail-based adaptivity of AWR.

Chapter 7

The Multiscale Adaptive Wavelet Basis

7.1 Introduction

The algorithms presented in chapters 5 and 6 perform well in most domains, but exhibit some deficiencies. AWR starts from a fixed basis, and so the size of the initial basis function set is exponential in the number of dimensions. IBFDD is unstable when the initial level of detail is insufficient. In this chapter, we present a hybrid algorithm which retains the theoretical properties of both AWR and IBFDD, and corrects the weaknesses of both approaches.

7.2 Multiscale Adaptive Wavelet Basis Algorithm

Our algorithm (MAWB) is an extension of IFDD to arbitrary basis functions in continuous domains, combined with an algorithm based on ATC which uses an IFDD-like relevance measure to decide when and where to split wavelet tiles. Our method uses an initial basis function set linear in the number of dimensions and which is insensitive to the initial scale of the basis functions, as it can adaptively add detail as necessary.

Algorithm 6 gives a complete overview of our algorithm which is intended to be independent of learning method or wavelet family. We have implemented ours using order 2 (quadratic) BSpline wavelets normalised to have a unit square integral, using Sarsa. We begin the algorithm with an initial wavelet basis Φ_0 at scale j, and supply the algorithm with tolerances τ_s and $\tau_c \geq 0$. At each timestep t, the algorithm draws a sample from the MDP, conducts learning on the weights w_t of the current basis function set Φ_t, and then updates the relevances of the currently activated candidate conjunctions and splits. If a candidate function is not currently activated, it has a value of zero in state s_t and cannot contribute in any way to the currently observed Bellman error δ_t. Once the relevances of each candidate conjunction and split are updated, we find the candidate conjunction and split with the largest magnitude relevance greater than tolerance τ (where the tolerances applied to IBFDD and AWR may differ). If we find a conjunctive candidate function with relevance greater than τ_c, it is added to the basis function set and pairwise conjunctions of itself and the other initial wavelets and candidate conjunctions are added to the set of candidate conjunctions, taking care not to add a candidate conjunction already added to either the set of basis functions or the set of candidate conjunctions. If we find a wavelet tile to split, the parent tile is replaced with its children by initialising their weights such that the value function remains unchanged, and potential splits of the children in each dimension are considered as potential splits. If no relevance is larger than τ_c or τ_s, the algorithm makes no change to the set of basis functions.

Algorithm 6 MAWB

```
Require: Intial wavelet basis scale $j$, Tolerances $\tau_s$ and $\tau_c$, $\gamma$ and the size of the domain
    of the function $\phi$ within the area of interest, $|\Omega_\phi|$
  Initialise basis function set $\Phi_0$ at scale $j$ such that each wavelet is along a state space
  axis.
  Initialise set of candidate conjunctions $\Phi_C$ through pairwise conjunctions of basis ele-
  ments in $\Phi_0$ such that the dimensions of each element in a pair are disjoint.
  Initialise $\rho(\phi) = 0$ for all $\phi \in \Phi_C$
  Initialise $\rho(\phi, d) = 0$ for all $\phi \in \Phi_0$ and $d \in [0, D]$
  Initialise $O(\phi) = 0$ for all $\phi \in \Phi_0$
  Initialise sample count $T(\phi) = 0$ for all $\phi \in \Phi_0$ and all $\phi \in \Phi_C$
  for each episode do
    Draw sample $s$, $a$, $r$, $s'$, $a'$ from MDP.
    while $s'$ not terminal: do
      Compute TD error $\delta_t$
      Update weights $w$ of basis $\Phi_t$ using sample and learning method
      Run IBFDD using $\Phi_t$, $s$, $\delta_t$, $\rho$, $T$, $O$, $\tau_c$
      Run AWR using $\Phi_t$, $s$, $\delta_t$, $\rho$, $T$, $O$, $\tau_s$
      Draw sample $s$, $a$, $r$, $s'$, $a'$ from MDP.
    end while
  end for
```

One may choose to maintain the set of candidate conjunctions as a set made of all pairwise conjunctions of elements of **F** at all times, which would require propagating any split throughout the set of basis functions and conjunctions. We have chosen to do the least amount of work possible, examining conjunctions at the coarsest level of detail only, and adding detail via splits. Candidate conjunctions never involve splits, and may only be created using the initial wavelet set. As such, the set of candidate conjunctions is of finite size.

One may decide to add all basis functions with relevance greater than τ, rather than only the largest. This approach is followed by Geramifard *et al.* [2011]. By adding only the largest, however, the rate of growth of the basis function set is guaranteed to be at most linear in the number of steps taken. This also acts to reduce sensitivity to the tolerance.

7.3 Theoretical Results

We present a number of theoretical results of MAWB, proving maximal reduction of approximation error, equivalence to an OMP method and convergence.

Theorem 7.1. *Given a stationary policy inducing ergodic chains on an MDP, a collection of wavelet basis functions* $\mathbf{\Phi}$ *at any scale and tolerances* $\tau \to 0$*, selecting the candidate function with the largest magnitude relevance will result in selecting the candidate function that maximally reduces the Bellman error in the weighted 2-norm.*

Proof. As time tends to infinity, the relevance of each candidate function will tend towards the inner product $\langle \Delta, \phi \rangle_P$, the projection of the wavelet ϕ onto the Bellman error. This inner product is proportional to the weight of the best (in the least squares sense) approximation of Δ using ϕ, and the weighted 2-norm of Δ given by $\sqrt{\int_\Omega \Delta^2(s)P(s)ds}$ is equal to the square root of the sum of the squares of the weights of the expansion of Δ in the wavelet basis. The candidate basis function with the maximum weight is the candidate function which would maximally reduce the Bellman error in the weighted 2-norm. After a finite number of samples, the candidate function with the largest magnitude relevance would be the candidate function that would have the largest magnitude projection. □

Theorem 7.2. *Given a stationary policy inducing ergodic chains on an MDP, a collection of wavelet basis functions $\mathbf{\Phi}$ at any scale with tolerances $\tau \to 0$, and sufficient time such that by selecting the candidate function with the largest magnitude relevance one would maximally reduce the Bellman error, the selected candidate function would also maximally reduce the approximation error $||\mathbf{V} - \mathbf{\Phi}\underline{w}||_P$ in the weighted 2-norm once the weight of the added function has been learned.*

Proof. Scherrer [2010] gives the result that the Bellman error (in a 2-norm weighted by the state density) of an approximation of $\mathbf{V}$ forms an upper bound (to a constant) of the approximation error in the same norm. By adding the candidate function that will maximally reduce the Bellman error in the weighted norm, we maximally reduce the upper bound on the approximation error, and thus maximally reduce that error. □

Theorem 7.3. *The multiscale adaptive wavelet approximation algorithm is an online approximation of an orthogonal matching pursuit algorithm.*

Proof. Since the relevance measure is a Monte Carlo approximation of the inner product $\langle \Delta, \phi \rangle_P$, it is an estimate of the projection of candidate function ϕ onto the Bellman error surface Δ. Selecting the largest such relevance measure means the algorithm has the same behaviour as a *matching pursuit* method, using projection estimates computed in finite time. As the weights associated with the basis functions are subsequently shifted by the addition of the candidate function ϕ, the method is an online approximation of an *orthogonal matching pursuit* algorithm. □

Theorem 7.4. *Given a changing policy π that, through exploration, induces ergodic Markov chains, and a learning method that converges to an optimal weight for any added function ϕ, MAWB will converge to a point where the projected error $\langle \Delta, \phi \rangle$ is below the combining tolerance τ_c for all existing candidate functions ϕ, and the projected error $\langle |\Delta|, \phi \rangle$ is below the splitting tolerance τ_s for all ϕ. For $\tau_c \to 0$, $\tau_s \to 0$, this implies convergence to either $\delta(s) \to 0$.*

Proof. This theorem combines theorems 5.4 and 6.2. Since both the AWR and IBFDD components converge if we make the assumption that the underlying learning mechanism converges to the optimal weight for any added ϕ, the assumptions of both theorems are satisfied. Thus, MAWB converges. □

This first convergence proof combines the convergence results of the two component algorithms of MAWB. Both convergence results rely on the convergence of the underlying learning method under the addition of any function to the basis function set, and thus neither algorithm can affect the convergence of the other. This rather stringent assumption is necessary, as Sarsa (and many other learning methods) do not have convergence results for value function approximation. If the learning method does not converge, the adaptive methods will not converge either.

Theorem 7.5. *Suppose we are dealing with an MDP with state variable x. Then for all $\epsilon > 0$ there exists an expanded wavelet basis $\Phi(x)$ and $\underline{w}$ such that $||V^*(x) - \underline{\Phi}(x)\underline{w}|| \leq \epsilon$ which can be reached from any intial basis scale for $\tau \to 0$.*

Proof. Since the algorithm is an online approximation of a matching pursuit method, it will converge in the space spanned by the basis functions. The space of conjunctions is fully explored by the IBFDD component of the algorithm, and the scale of each basis function grows as those functions are split. Thus, the space spanned by the basis functions tends towards $\mathcal{L}_2(\mathbb{R}^n)$, and the algorithm converges in that space. □

This second convergence proof utilises the relationship between MAWB and OMP methods, and implies the existence of a wavelet basis with arbitrarily low approximation error that can be reached from any initial basis, with appropriately chosen tolerances tending to zero. In practice, this *best basis* may not be possible to reach due to time limitations, as the theorem provides no guarantee on the speed of convergence.

7.4 Empirical Results

The IBFDD method was tested against MAWB using order 2 B-spline wavelets, at scales 1 and 2 (or equivalently 4 or 6 functions per dimension). We use Sarsa(λ) with $\lambda = 0.9$, $\epsilon = 0.05$ and $\gamma = 1$. PARL2 α scaling was used [Dabney 2014], with $\alpha_0 = 1$. Results were averaged across 100 experiments for 20 adaptive episodes followed by 80 nonadaptive episodes, in order to demonstrate long-term performance of the learned basis. All results show the average return measured at each episode, and the average number of basis functions in the basis function set at each episode. The tolerances were chosen to match the tolerances used in the results of Chapters 5 and 6.

7.4.1 Discontinuous Room

Figures 7.1 and 7.2 show that MAWB does not suffer the same instability as IBFDD when the initial set of features is undetailed. We also see that when the initial feature set is sufficiently detailed, not many basis functions are added (scale 2).

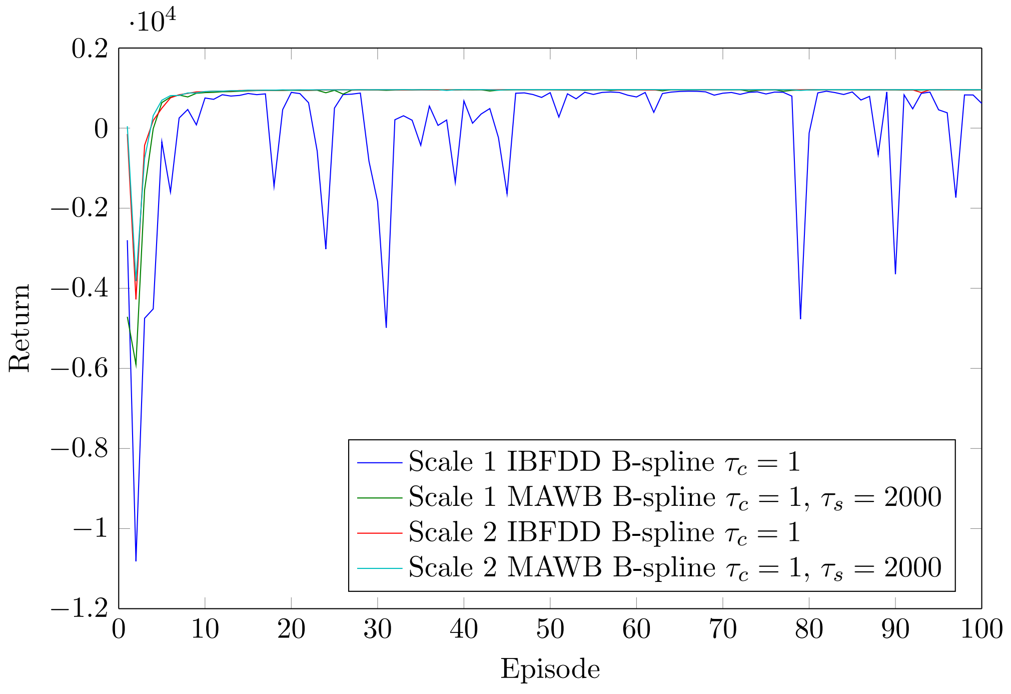


Figure 7.1: MAWB Returns for Discontinuous Room

7.4.2 Mountain Car

Figures 7.3 and 7.4 show again that MAWB is more stable than IBFDD. Additionally, we see that MAWB achieves better results than IBFDD, although more basis functions are used to achieve this. Equivalent tolerances result in fewer splits taking place at higher scales, as expected.

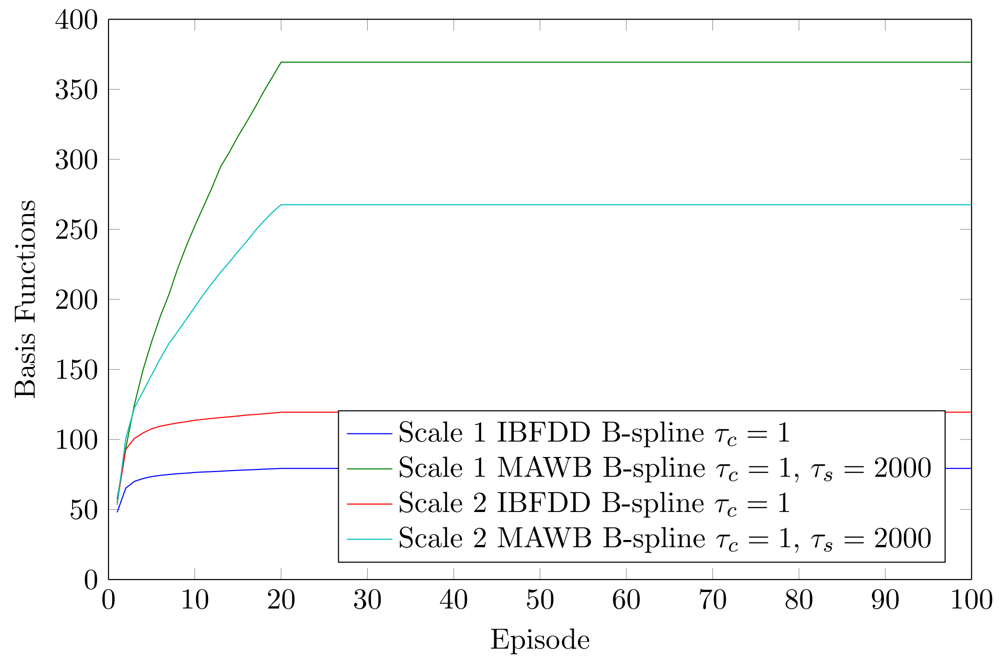


Figure 7.2: MAWB Number of Basis Functions for Discontinuous Room

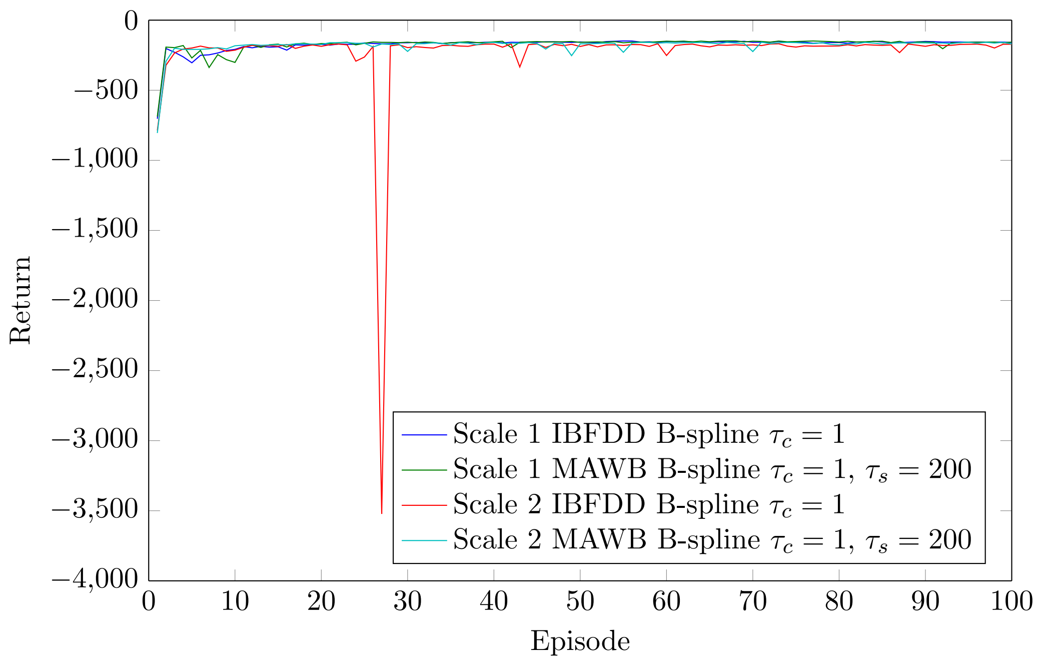


Figure 7.3: MAWB Returns for Mountain Car

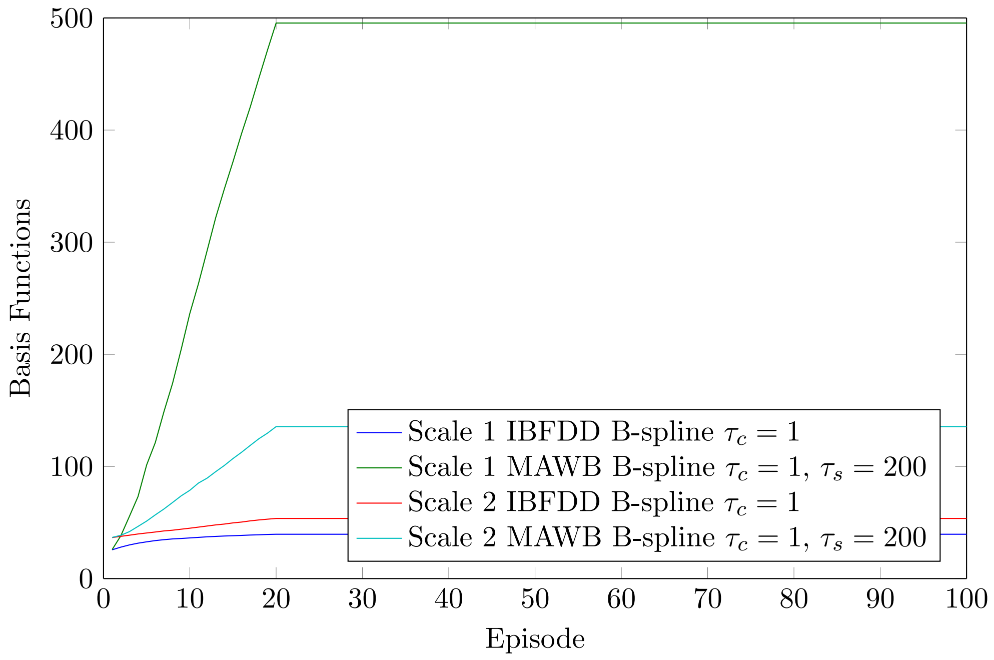


Figure 7.4: MAWB Number of Basis Functions for Mountain Car

7.4.3 Acrobot

Figures 7.5 and 7.6 show that, even in MAWB, B-splines outperform tile coding. The results of MAWB are better than those of IBFDD for both basis function types. These results are consistent for both starting scales (figures 7.7 and 7.8).

7.4.4 Pinball

Figures 7.9 and 7.10 show that, again, MAWB has corrected an instability present in IBFDD, resulting in better performance.

7.4.5 3D Mountain Car

For B-splines at scale 1, no appreciable difference in performance between MAWB and IBFDD is observed, but at scale 2, we see that MAWB outperforms IBFDD, resulting in better, and more stable, performance (figures 7.11 and 7.12).

7.4.6 Car Driving Simulation

We present the n-lane Car Driving Simulation domain to demonstrate further the performance of MAWB. This domain is based on the 3 lane Car Driving Simulation domain commonly used in inverse reinforcement learning, extended to n lanes. We use as a state space the distance to the nearest car for each lane, and the lane the agent is currently driving in (for a total of $n + 1$ continuous state dimensions). We employ the *nice* reward structure described by Abbeel and Ng [2004]. Figures 7.13 and 7.14 show the results of employing scale 0, 1 and 2 B-splines in a 19 lane simulation for IBFDD and MAWB. We see that for scales 0 and 1, MAWB clearly outperforms IBFDD, while for scale 2, it appears that MAWB performs better for most episodes, but not substantially.

Despite the high dimensionality of the state space (20 dimensions), the number of basis functions in the basis function set remains low, with less than 6000 basis functions being employed. If we compare this to a fixed basis, we would require the use of fewer than 1.6

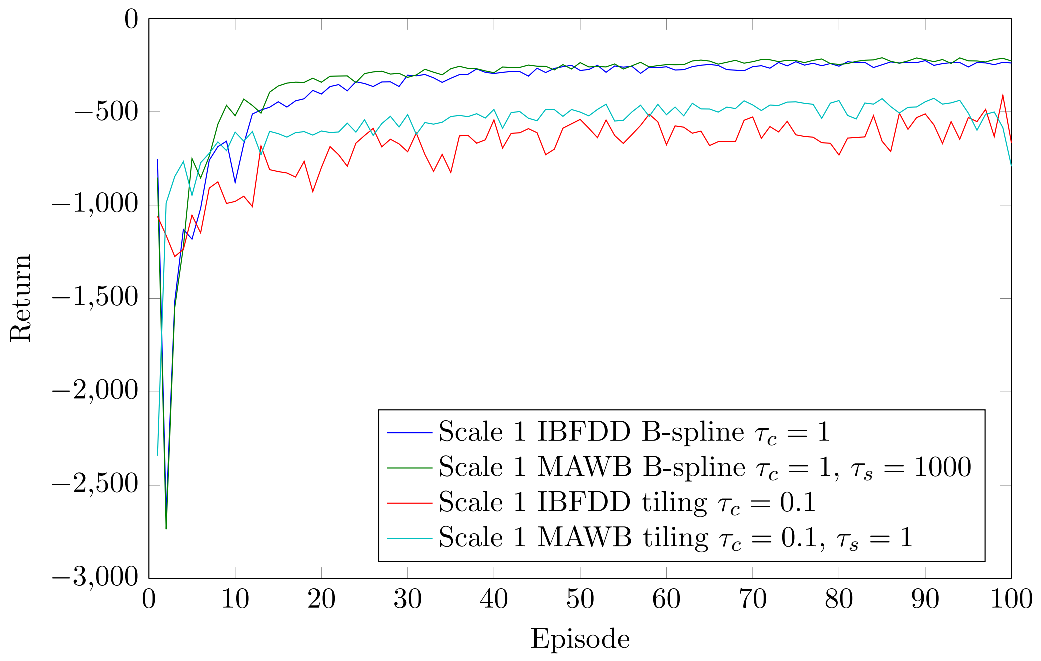


Figure 7.5: MAWB Returns for Acrobot at Scale 1

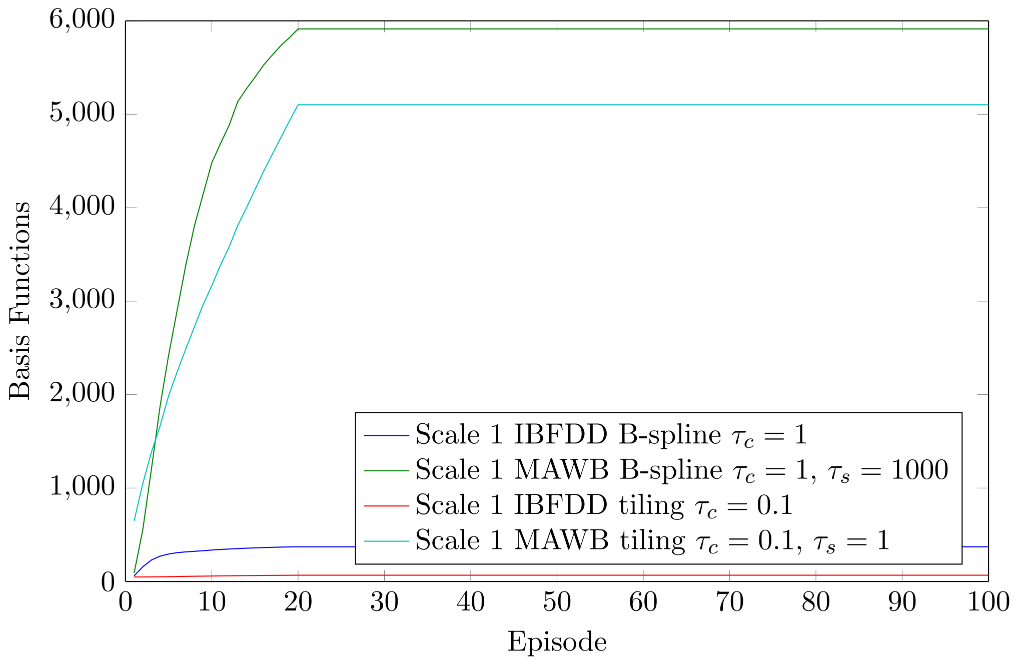


Figure 7.6: MAWB Number of Basis Functions for Acrobot at Scale 1

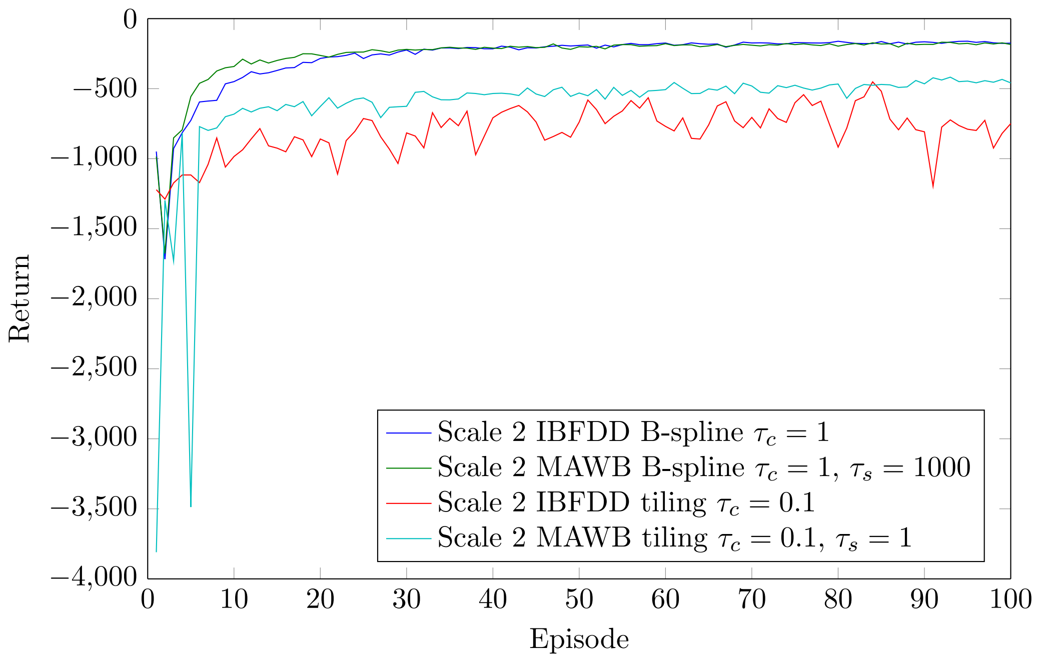


Figure 7.7: MAWB Returns for Acrobot at Scale 2

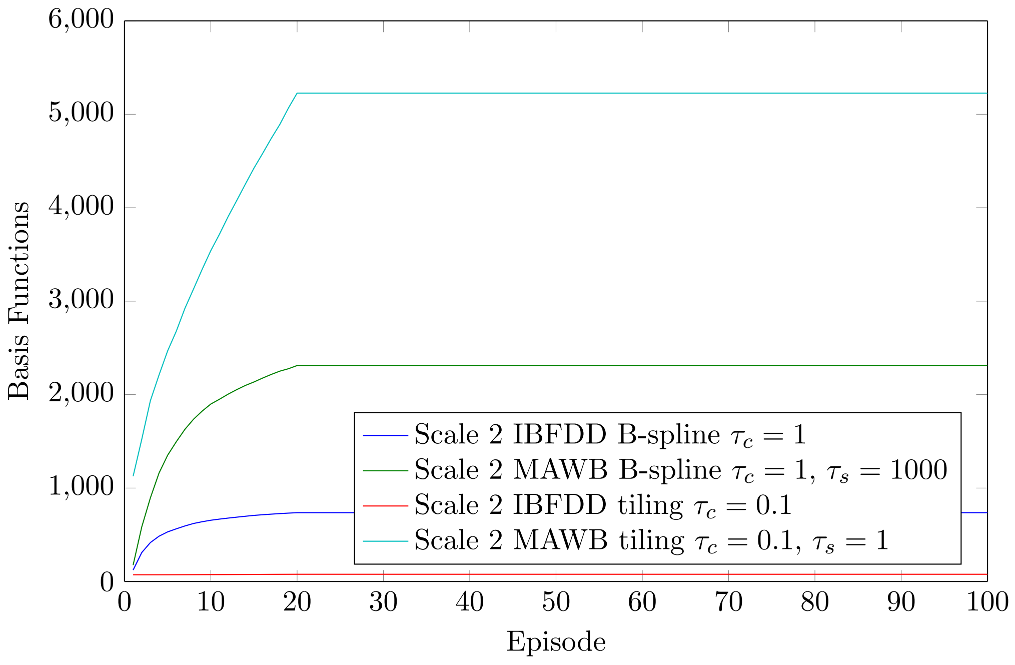


Figure 7.8: MAWB Number of Basis Functions for Acrobot at Scale 2

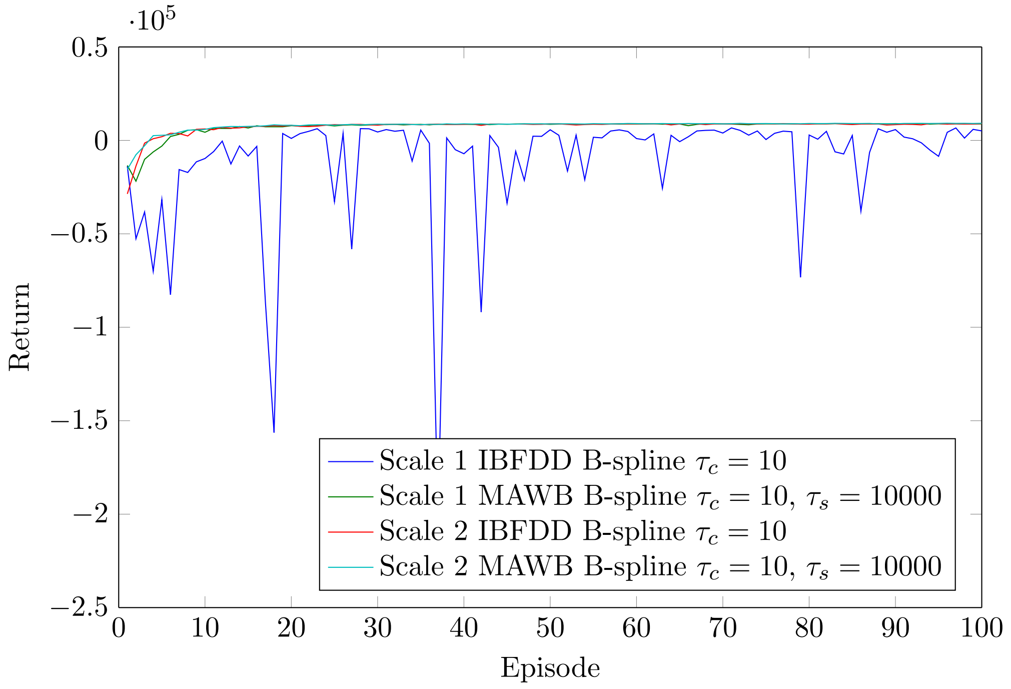


Figure 7.9: MAWB Returns for Pinball

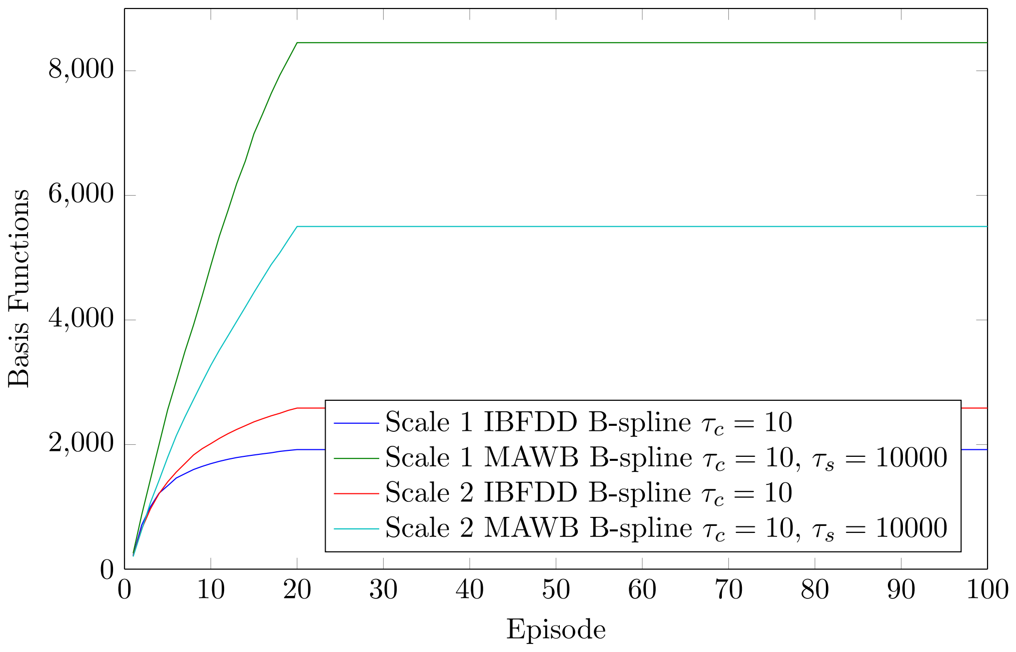


Figure 7.10: MAWB Number of Basis Functions for Pinball

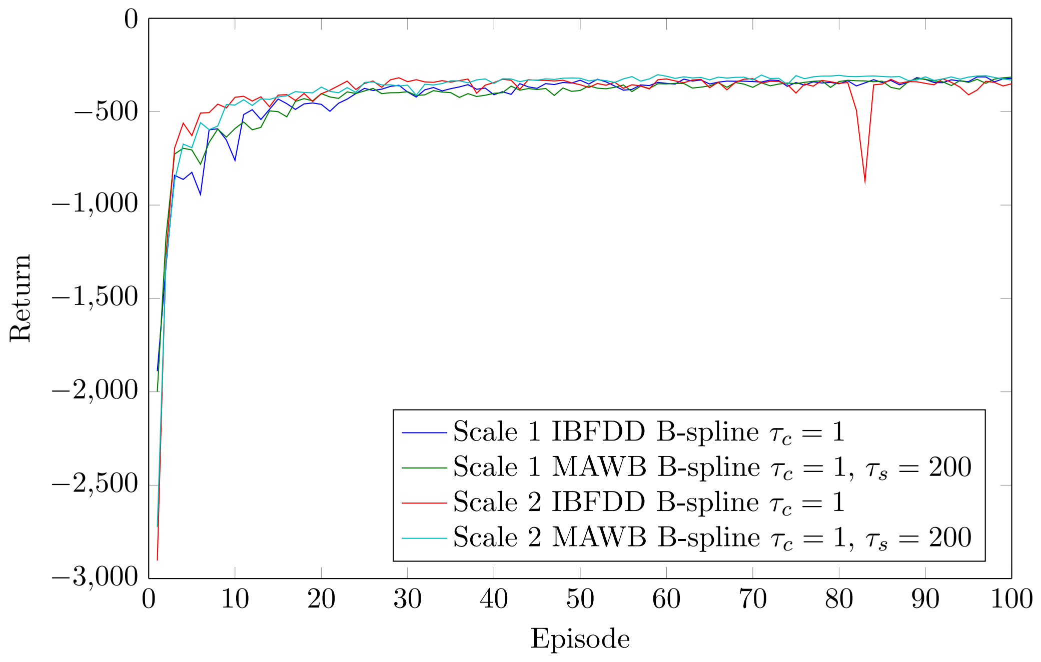


Figure 7.11: MAWB Returns for 3D Mountain Car

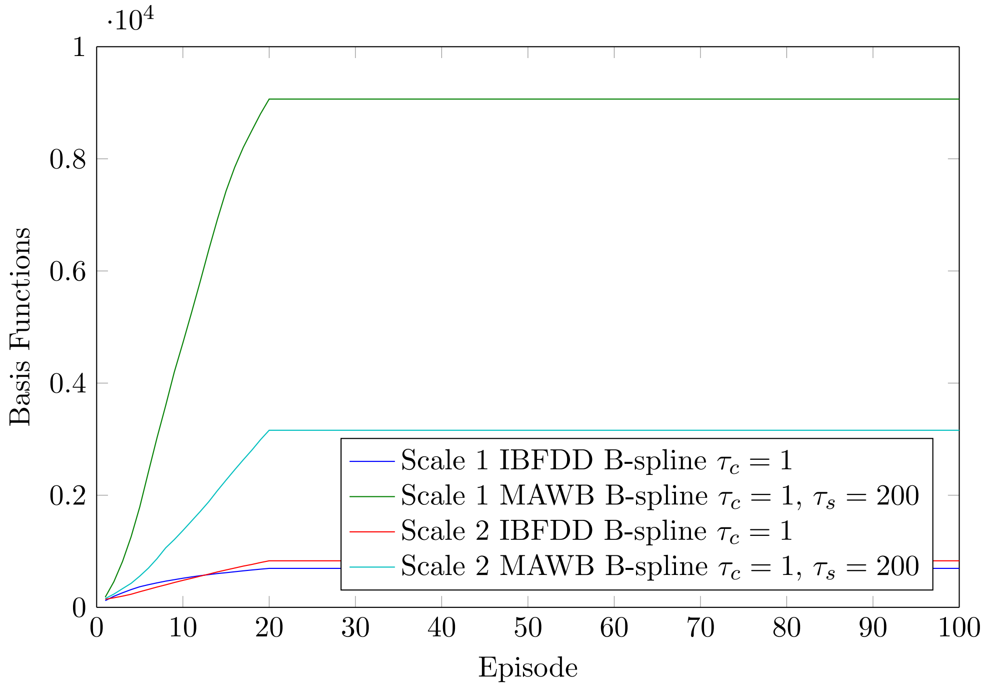


Figure 7.12: MAWB Number of Basis Functions for 3D Mountain Car

basis functions per dimension to achieve a basis function set of that size (using two basis functions per dimension would result in a basis function set employing 1 048 576 basis functions).

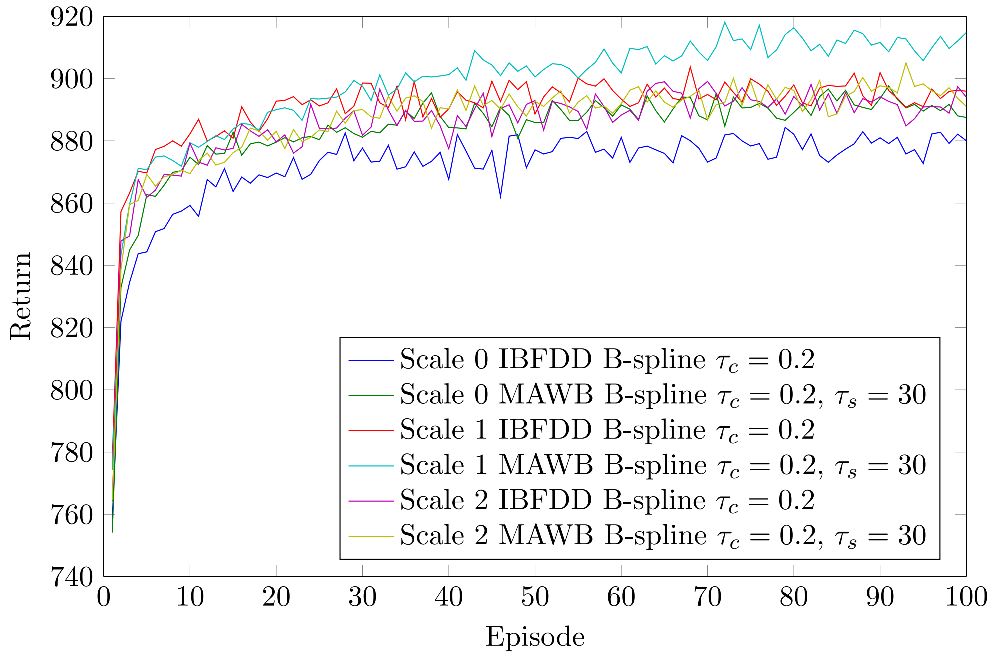


Figure 7.13: MAWB Returns for 19 lane Car Task

To demonstrate the dimensional scalability of MAWB, we ran the car driving simulation from 9 to 99 lanes (10 to 100 dimensions) in steps of ten for 20 episodes of 1000 steps each using a second order B-spline MAWB begun at scale 0. The total compute time and final return at episode 20 were recorded for each, and the results were averaged across 10 experiments. The simulations were run on a laptop with 16gb memory and 8 CPU cores, with no special care taken to provide speed advantages, no threading and standard hardware. The average times taken for each dimension are shown in figure 7.15. We see that even the 100 dimensional domain is run in a reasonable timeframe, and the rate of growth appears to be quadratic (although the difficulty in computing the average frontier size means this is a qualitative observation only). Furthermore, we can show that this subexponential growth does not come at a cost of unduly reduced performance (some performance loss is expected, as the portion of the state space explored through the fixed number of samples shrinks exponentially). In figure 7.16, we see that the final returns obtained by MAWB do not change drastically.

Were one to attempt the 100 dimensional task with a fixed basis at the same detail level, one would require 3^{100} basis functions, a number rather close to the number of atoms comprising the planet.

7.5 Discussion

The results of this chapter are summarised in table 7.1. We observe that the use of MAWB has resulted in better performance than IBFDD in all domains. The application of MAWB to a very high dimensional problem was also demonstrated using the car driving simulation up to 100 dimensions.

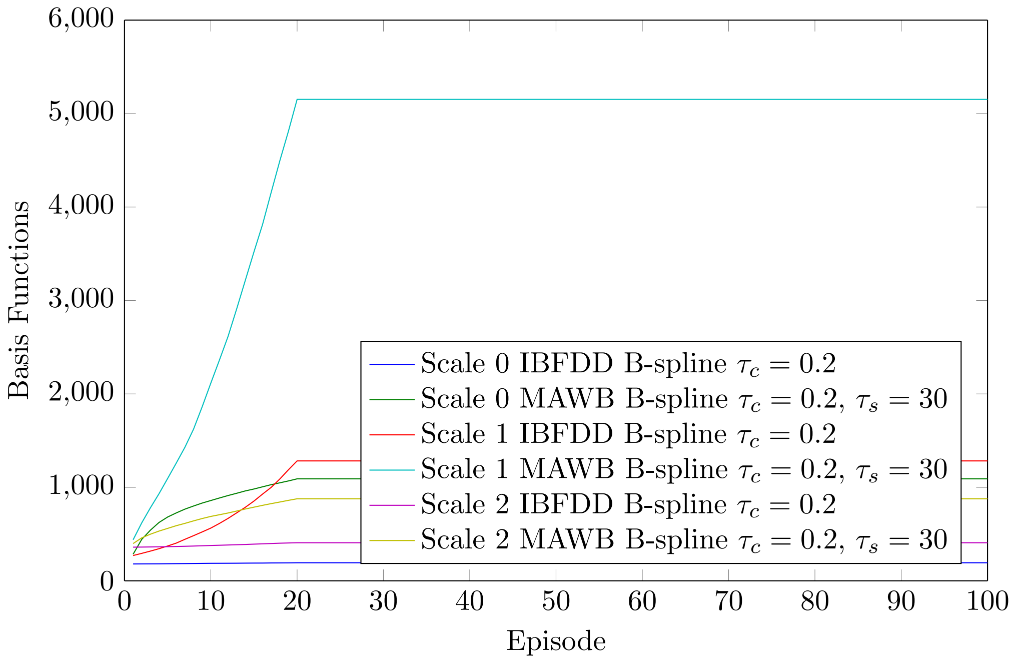


Figure 7.14: MAWB Number of Basis Functions for 19 lane Car Task

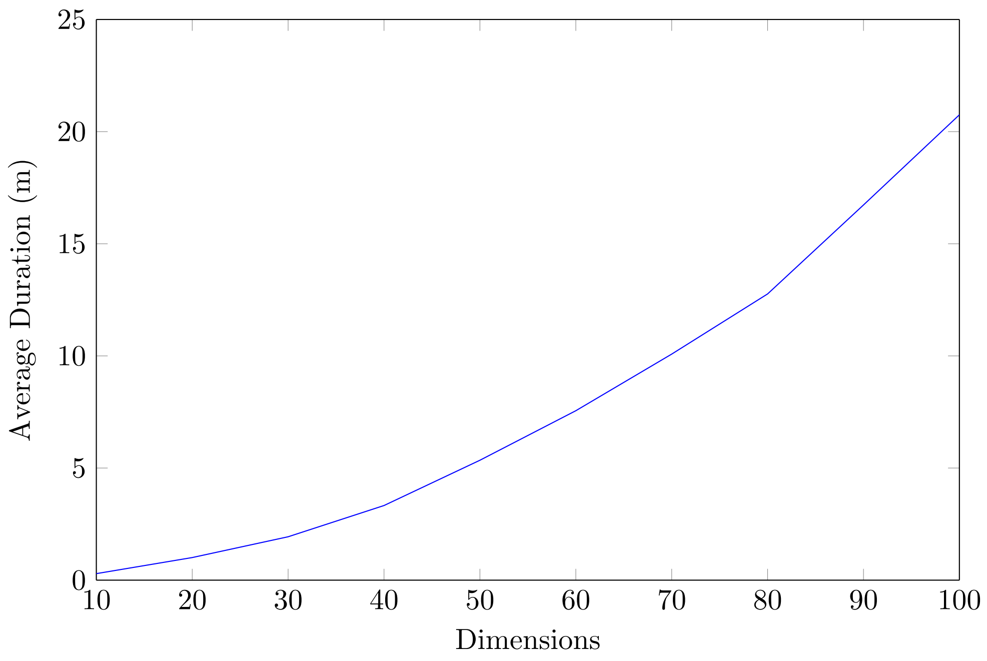


Figure 7.15: Time taken to execute Car Tasks of varying dimensions

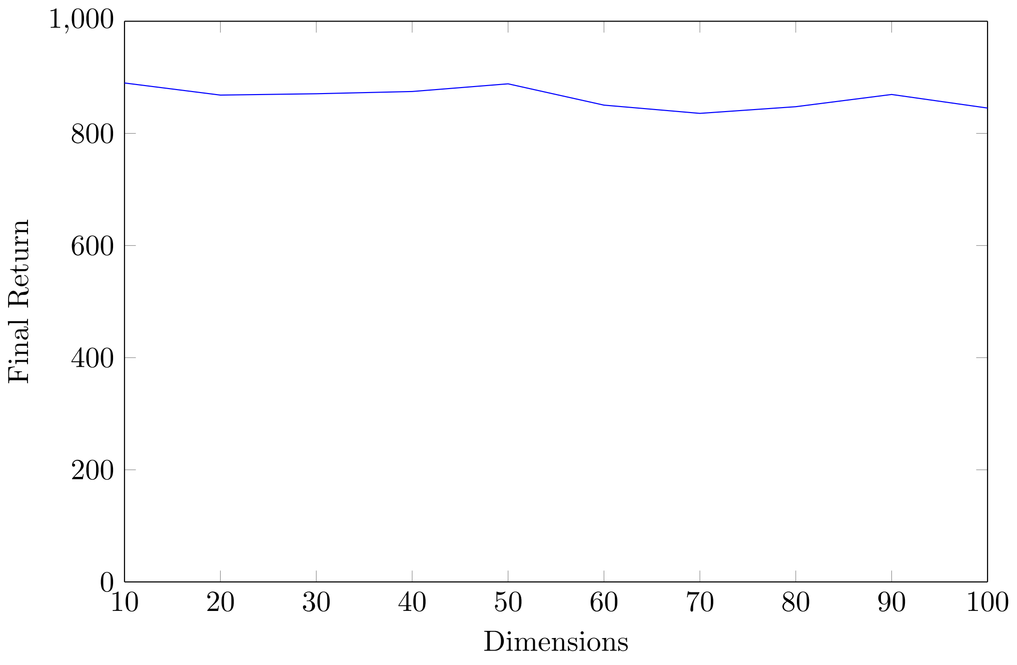


Figure 7.16: Final returns of Car Tasks of varying dimensions

	MAWB vs. IBFDD
Discontinuous Room	MAWB better
Mountain Car	MAWB better
Acrobot	MAWB better
Pinball	MAWB better
3D Mountain Car	MAWB better
Car Task	MAWB better

Table 7.1: MAWB results summary

7.6 Conclusion

The MAWB method is dimensionally insensitive (like IBFDD), and detail insensitive (like AWR). This means that domains of arbitrarily high dimension can be represented using low detail bases, since dimensional interdependency will be discovered with IBFDD, and required detail will be discovered with AWR. MAWB retains the convergence proofs of its two component algorithms, and has a further convergence property in being an online approximation to a matching pursuit method. MAWB has demonstrably better performance when compared to IBFDD, which has better performance than IFDD and IFDD+.

Chapter 8

Conclusion

The B-spline basis is shown to be competitive against other basis function types as a fixed basis, performing better than the state of the art in almost all tested domains. The performance of both the tile-based and the wavelet-based AWR are better than or equivalent to the performance of a fixed basis. In some cases, the AWR performance exceeded that of a fixed basis at the next scale. Similarly, wavelet-based IBFDD performed better than IFDD and IFDD+. These results carried over to MAWB, wherein the performance of IBFDD was further improved by allowing wavelet splitting through AWR. MAWB is scalable to high dimensional problems, with performance shown up to 100 dimensions.

8.1 Contributions

This thesis makes five main contributions. The first is a clear explanation and demonstration of the usage of wavelets for function synthesis within the context of reinforcement learning. The second is a number of measures of the relevance or accumulated error attributable to a function in an online setting. The third is Adaptive Wavelet Refinement, an extension of the ATC algorithm [Whiteson *et al.* 2007] to arbitrary online learning methods using any refineable basis function, together with proofs of the necessity and sufficiency of wavelets as a basis function type, and a proof of convergence of the AWR algorithm. The fourth contribution is Incremental Basis Function Dependency Discovery, an extension of IFDD [Geramifard *et al.* 2011] to arbitrary basis function types, together with a proof of convergence of the IBFDD algorithm. The final contribution is a hybrid algorithm combining IBFDD and AWR into MAWB, a multiscale wavelet-based basis function scheme that scales to high dimensional state spaces and is detail and dimensionally adaptive.

8.2 Future work

While the novel measures and algorithms provide good performance, there are many design decisions to explore further. The relevance measures of Chapter 4 do not take into account function variance, which would likely provide more information about function stability and convergence than the mean. AWR uses the difference between the observed error and the relevance to decide which function to split, whereas the difference between measures of the parent and the child may provide better performance. Furthermore, while the weight of the child functions can be set to match that of the parent, there is no obvious way to capture the parent function's learning *trajectory* in the children, leading to slower, pessimistic learning. The candidate conjunction sets of IBFDD and MAWB are maintained as conjuctions of wavelets at the coarsest level of detail, and added functions are never removed, even if they are entirely covered by their conjunctive children. A better data structure would be required to maintain a smarter, more robust list of conjunctions, one that is perhaps made up of only the most *useful* functions, for some measure of usefulness.

It would furthermore be useful to investigate *why* some functions perform better in some domains, independently of the learning method. This may provide insight into how to choose a good basis from which to adapt. Ideally, one would have in the basis function set only those functions that are absolutely required for learning at the current detail level: learning an initial value function requires generalisation, while learning later requires the addition of detail. One might imagine an adaptive basis function scheme that, together with a well-chosen step-size and exploration parameter, guides learning through an optimal balance of generality and fine detail, while still being scalable to high dimensional problems. We hope the work in this thesis is a step in that direction, albeit one made with a small value of α.

Bibliography

[Abbeel and Ng 2004] P. Abbeel and A.Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

[Baird 1995] Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 30–37. Morgan Kaufmann, 1995.

[Berger and Colella 1989] M.J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82(1):64 – 84, 1989.

[Bishop and Miikkulainen 2013] Julian Bishop and Risto Miikkulainen. Evolutionary feature evaluation for online reinforcement learning. In *Computational Intelligence in Games, 2013 IEEE Conference on*, pages 1–8. IEEE, 2013.

[Bowling *et al.* 2008] Michael Bowling, Alborz Geramifard, and David Wingate. Sigma point policy iteration. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 379–386. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[Boyan 1999] J.A. Boyan. Least squares temporal difference learning. In *Proceedings of the 16th International Conference on Machine Learning*, pages 49–56, 1999.

[Cho and Lai 2005] Okkyung Cho and Ming Jun Lai. A class of compactly supported orthonormal B-spline wavelets. *Splines and Wavelets*, 2005.

[Christensen 2008] O. Christensen. *Frames and Bases.* Birkhauser Boston, 2008.

[Chui and Wang 1992] Charles K Chui and Jian-zhong Wang. On compactly supported spline wavelets and a duality principle. *Transactions of the American Mathematical Society*, 330(2):903–915, 1992.

[Chui 1997] C.K. Chui. *Wavelets: a Mathematical Tool for Signal Analysis.* SIAM, 1997.

[Čisar and Čisar 2011] Petar Čisar and Sanja Maravić Čisar. Optimization methods of EWMA statistics. *Acta Polytechnica Hungarica*, 8(5):73–87, 2011.

[Coifman and Maggioni 2006] Ronald R Coifman and Mauro Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1):53–94, 2006.

[Dabney 2014] William Dabney. *Adaptive Step-Sizes for Reinforcement Learning.* PhD thesis, University of Massachusetts Amherst, 2014.

[Daubechies 1992] I. Daubechies. *Ten Lectures on Wavelets.* SIAM, 1992.

[Fard *et al.* 2013] Mahdi Milani Fard, Yuri Grinberg, Amir Massoud Farahmand, Joelle Pineau, and Doina Precup. Bellman error based feature generation using random projections on sparse spaces. In *Advances in Neural Information Processing Systems*, pages 3030–3038, 2013.

[Främling 2007] Kary Främling. Replacing eligibility trace for action-value learning with function approximation. In *European Symposium on Artificial Neural Networks*, pages 313–318, 2007.

[Ganesan *et al.* 2007] R. Ganesan, T.K. Das, and K.M. Ramachandran. A multiresolution analysis-assisted reinforcement learning approach to run-by-run control. *Automation Science and Engineering, IEEE Transactions on*, 4(2):182 –193, April 2007.

[Geramifard *et al.* 2011] A. Geramifard, F. Doshi-Velez, J. Redding, N. Roy, and J. How. Online discovery of feature dependencies. In *Proceedings of the 28th International Conference on Machine Learning*, pages 881–888, New York, NY, USA, June 2011.

[Geramifard *et al.* 2013a] Alborz Geramifard, Christoph Dann, and Jonathan P How. Off-policy learning combined with automatic feature expansion for solving large MDPs. In *Proc. 1st Multidisciplinary Conf. on Reinforcement Learning and Decision Making*, pages 29–33, 2013.

[Geramifard *et al.* 2013b] Alborz Geramifard, Thomas J Walsh, Nicholas Roy, and Jonathan How. Batch iFDD: A scalable matching pursuit algorithm for solving MDPs. In *Proceedings of the 29th Annual Conference on Uncertainty in Artificial Intelligence*, 2013.

[Hansen *et al.* 2015] EC Hansen, A Frank, and P Hartigan. Magnetohydrodynamic effects on pulsed young stellar object jets. i. 2.5 d simulations. *The Astrophysical Journal*, 800:41, 2015.

[Kaiser 2010] Gerald Kaiser. *A friendly guide to wavelets.* Springer, 2010.

[Keinert 2003] Fritz Keinert. *Wavelets and multiwavelets.* CRC Press, 2003.

[Keller *et al.* 2006] P.W. Keller, S. Mannor, and D. Precup. Automatic basis function construction for approximate dynamic programming and reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, pages 449–456, 2006.

[Kolter and Ng 2009] J.Z. Kolter and A.Y. Ng. Regularization and feature selection in least-squares temporal difference learning. In *Proceedings of the 26th International Conference on Machine Learning*, pages 521–528, 2009.

[Konidaris and Barto 2009] G.D. Konidaris and A.G. Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in Neural Information Processing Systems 22*, pages 1015–1023, 2009.

[Konidaris *et al.* 2011] G.D. Konidaris, S. Osentoski, and P.S. Thomas. Value function approximation in reinforcement learning using the Fourier basis. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence*, pages 380–385, 2011.

[Konidaris 2011] G.D. Konidaris. *Autonomous Robot Skill Acquisition.* PhD thesis, University of Massachusetts Amherst, May 2011.

[Kritchman and Nadler 2008] Shira Kritchman and Boaz Nadler. Determining the number of components in a factor model from limited noisy data. *Chemometrics and Intelligent Laboratory Systems*, 94(1):19–32, 2008.

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

[Lin and Wright 2010] Stephen Lin and Robert Wright. Evolutionary tile coding: An automated state abstraction algorithm for reinforcement learning. In *Abstraction, Reformulation, and Approximation*, 2010.

[Maei and Sutton 2010] Hamid Reza Maei and Richard S Sutton. GQ (λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces. In *Proceedings of the Third Conference on Artificial General Intelligence*, volume 1, pages 91–96, 2010.

[Maei *et al.* 2010] Hamid Reza Maei, Csaba Szepesvari, Shalabh Bhatnagar, and Richard S. Sutton. Toward off-policy learning control with function approximation. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning*, pages 719–726, June 2010.

[Mahadevan and Maggioni 2006] S. Mahadevan and M. Maggioni. Value function approximation with diffusion wavelets and Laplacian eigenfunctions. *Advances in Neural Information Processing Systems 18*, pages 843–850, 2006.

[Mahadevan 2005] S. Mahadevan. Proto-value functions: Developmental reinforcement learning. In *Proceedings of the Twenty Second International Conference on Machine Learning*, pages 553–560, 2005.

[Mairal *et al.* 2008] Julien Mairal, Marius Leordeanu, Francis Bach, Martial Hebert, and Jean Ponce. Discriminative sparse image models for class-specific edge detection and image interpretation. In *Computer Vision ECCV 2008*, volume 5304 of *Lecture Notes in Computer Science*, pages 43–56. Springer Berlin Heidelberg, 2008.

[Melosh and Killian 1976] RJ Melosh and DE Killian. Adaptive mesh refinement in finite element analysis. In *Second National Symposium on Computerized Structural Analysis and Design, Washington, DC*, 1976.

[Menache *et al.* 2005] Ishai Menache, Shie Mannor, and Nahum Shimkin. Basis function adaptation in temporal difference reinforcement learning. *Annals of Operations Research*, 134(1):215–238, 2005.

[Mnih *et al.* 2013] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[Mohlenkamp and Pereyra 2008] Martin J Mohlenkamp and María Cristina Pereyra. *Wavelets, their friends, and what they can do for you.* European Mathematical Society, 2008.

[Munos and Moore 2000] Rémi Munos and Andrew W. Moore. Rates of convergence for variable resolution schemes in optimal control. In *Proceedings of the Seventeenth International Conference on Machine Learning, Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, pages 647–654, 2000.

[Munos and Moore 2002] Rémi Munos and Andrew Moore. Variable resolution discretization in optimal control. *Machine learning*, 49(2-3):291–323, 2002.

[Painter-Wakefield and Parr 2012] C. Painter-Wakefield and R. Parr. Greedy algorithms for sparse reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, pages 1391–1398, 2012.

[Parr *et al.* 2007] Ronald Parr, Christopher Painter-Wakefield, Lihong Li, and Michael Littman. Analyzing feature generation for value-function approximation. In *Proceedings of the 24th international conference on Machine learning*, pages 737–744. ACM, 2007.

[Parr *et al.* 2008] Ronald Parr, Lihong Li, Gavin Taylor, Christopher Painter-Wakefield, and Michael L Littman. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 752–759. ACM, 2008.

[Petrik 2007] Marek Petrik. An analysis of Laplacian methods for value function approximation in MDPs. In *International Joint Conferences on Artificial Intelligence*, pages 2574–2579, 2007.

[Pujol 2007] Gerard Llort Pujol. *Improvement of the Spatial Resolution for Multibeam Echosounders.* PhD thesis, Universite de Rennes I, 2007.

[Razo-Zapata *et al.* 2007] I.S. Razo-Zapata, J. Waissman-Vilanova, and L.E. Ramos-Velasco. Reinforcement learning in continuous systems: Wavelet networks approach. In *Analysis and Design of Intelligent Systems using Soft Computing Techniques*, volume 41 of *Advances in Soft Computing*, pages 727–736. Springer Berlin Heidelberg, 2007.

[Scherrer 2010] Bruno Scherrer. Should one compute the temporal difference fix point or minimize the bellman residual? The unified oblique projection view. In *Proceedings of the 27th International Conference on Machine Learning*, pages 959–966, 2010.

[Soman *et al.* 2009] K.P. Soman, T. Arathi, M.S. Augustine, and S.V. Arunima. Daubechies-Lagarias algorithm — a simplified approach. In *Proceedings of the 2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies*, pages 510–512, 2009.

[Strang 1989] G. Strang. Wavelets and dilation equations: A brief introduction. *SIAM Rev.*, 31(4):614–627, December 1989.

[Strela 1996] Vasily Strela. *Multiwavelets: theory and applications.* PhD thesis, Massachusetts Institute of Technology, 1996.

[Sutton and Barto 1998] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, 1998.

[Taylor *et al.* 2008] M.E. Taylor, G. Kuhlmann, and P. Stone. Autonomous transfer for reinforcement learning. In *The Seventh International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 283–290, 2008.

[Triebel 2008] H. Triebel. *Function spaces and wavelets on domains.* European Mathematical Soc.,, 2008.

[Tsitsiklis and Van Roy 1997] J.N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *Automatic Control, IEEE Transactions on*, 42(5):674–690, May 1997.

[Unser *et al.* 1993] Michael Unser, Akram Aldroubi, and Murray Eden. A family of polynomial spline wavelet transforms. *Signal processing*, 30(2):141–162, 1993.

[Unser 1997] Michael A Unser. Ten good reasons for using spline wavelets. In *Optical Science, Engineering and Instrumentation*, pages 422–431. International Society for Optics and Photonics, 1997.

[Watkins and Dayan 1992] C.J.C.H Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

[Whiteson and Stone 2006] Shimon Whiteson and Peter Stone. Evolutionary function approximation for reinforcement learning. *The Journal of Machine Learning Research*, 7:877–917, December 2006.

[Whiteson *et al.* 2007] S. Whiteson, M.E. Taylor, and P. Stone. *Adaptive tile coding for value function approximation.* Technical report, Computer Science Department, University of Texas at Austin, 2007.

[Wu and Meleis 2009] Cheng Wu and Waleed Meleis. Function approximation using tile and Kanerva coding for multi-agent systems. In *Proceedings of the Adaptive Learning Agents Workshop*, 2009.