Environmental Modelling & Software 38 (2012) 1–12

Contents lists available at SciVerse ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft

Comparing approximation techniques to continuous-time stochastic dynamic programming problems: Applications to natural resource modelling

Tom Kompas*, Long Chu

Crawford School of Economics and Government, Building 132, Lennox Crossing, Australian National University, Canberra ACT 0200, Australia

ARTICLE INFO

Article history:
Received 16 September 2011
Received in revised form
1 April 2012
Accepted 10 April 2012
Available online xxx

JEL classification:
C61
C63
Q22

Keywords:
Continuous-time stochastic dynamic programming
Parametric approximation
Perturbation method
Projection technique
Linear programming
Natural resource management
Fisheries management
Marine reserves

ABSTRACT

Dynamic programming problems are common in economics, finance and natural resource management. However, exact solutions to these problems are exceptional. Instead, solutions typically rely on numerical approximation techniques which vary in use, complexity and computational requirements. Perturbation, projection and linear programming approaches are among the most useful of these numerical techniques. In this paper, we extend the parametric linear programming technique to include continuous-time problems with jump-diffusion processes, and compare it to projection and perturbation techniques for solving dynamic programming problems in terms of computational speed, accuracy, ease of use and scope. The comparisons are drawn from solutions to two fisheries management problems – a unidimensional model of optimal harvest and a multidimensional model for optimal marine reserve size. Available computer code illustrates how each technique solves these problems and how they can be applied to other comparable problems in natural resource modelling.

1. Introduction

Stochastic optimal control problems are used extensively in economics, finance and natural resource modelling. These problems normally involve optimising a life-time (discounted) value functional, subject to an allowable transition of state variables that can be controlled by a decision maker. Time in these problems can be modelled as a continuous or discrete variable, with the transition state governed by either differential or difference equations. The choice of a discrete or continuous-time framework depends on the setting and is usually determined on a case-by-case basis (Doraszelski and Judd, 2010). In many economic problems, modelling time as a discrete or continuous variable does not alter any of the important qualitative properties of the model, hence the choice of the time framework mainly depends on the convenience it provides to modellers, and occasionally on the data available to support the model.

However, in other situations, the choice between a discrete or continuous setting is crucial. For example, a discrete-time framework is more appropriate for seasonal, delayed or lagged effects, as illustrated in Clark (1976). On the other hand, modelling a diffusion process from a high density source to a low density sink in discrete-time may cause 'unexpected overshooting', i.e., after one period of time the diffusion process may make the sink more dense than the source. In this case, a continuous-time framework is more appropriate since the gradual adjustment over time will prevent such cases. In problems where discrete-time approximations do work well, and where 'overshooting' is not an issue, extensions to large state-spaces (e.g., Nicol and Chades, 2010) can generate a significant advantage over continuous-time methods.

Both discrete and continuous-time problems can be approached with the 'Principle of Optimality' introduced by Bellman (1957). In discrete-time, the Principle of Optimality leads to a so-called Bellman equation. In a continuous-time framework, the Principle of Optimality leads to Hamiltonian-Jacobi-Bellman (HJB) equation.

* Corresponding author.
E-mail addresses: tom.kompas@anu.edu.au (T. Kompas), long.chu@anu.edu.au (L. Chu).

1364-8152/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.envsoft.2012.04.002

Bellman and HJB equations can generate exact solutions in only a very few cases and normally solutions must instead rely on approximation techniques.

In general, the approximation techniques for HJB equations are more restrictive than for Bellman equations. The most common techniques to approximate a Bellman equation are value and policy function iterations. These are well-established and stable techniques which rely on the convergence of an iteration process. However, they do not work well for continuous-time problems. Theoretically, one can model continuous-time as a 'discretized-time' problem, with very small time intervals, and the use of value or policy function iterations. Unfortunately, as is well-known and detailed in Judd (1998) and Doraszelski and Judd (2010), the appropriate discount factor will then be close to unity causing the iteration process to converge very slowly.

Solving a HJB equation numerically usually relies on parametric approximation techniques which try to produce an analytical formula in the form of a linear combination of some pre-determined basis functions.[1] Three parametric techniques which have been applied to optimal control problems are the perturbation, projection and linear programming techniques. The perturbation and projection methods are well-known in economics due to the works of Judd and Guu (1993), Judd (1996, 1998), Gaspar and Judd (1997) and Arruda and DoVal (2006). Linear programming approaches to dynamic problems attract far less attention, and their application to HJB equations has only recently been introduced by the mathematicians, Han and Roy (2009), although the original idea can be dated back to Manne (1960) and Ross (1970).

While there are at least two papers that compare various numerical methods in discrete-time dynamic programming, namely Taylor and Uhlig (1990) and Aruoba et al. (2006), comparisons of techniques for continuous-time problems are scant and usually brief. Gaspar and Judd (1997), for example, briefly indicate that the projection technique is typically much slower than a perturbation method, while Judd (1998) simply points out that the projection technique is a general approach to functional equations, hence naturally applicable to continuous-time dynamic programming problems. Han and Roy (2009), while introducing the basic structure of the linear programming technique, do not explore computational issues, and only mention that this new technique is efficient and convenient.

In this paper, we compare the three parametric techniques for HJB equations to highlight their relative strengths and weaknesses to continuous-time problems with jump-diffusion processes – processes especially applicable to problems in natural resource modelling. Our comparison differs from the two previous works that compare these techniques for discrete-time problems in at least three ways. First, the comparison of the techniques' performance is in an applied framework, illustrated through two useful numerical case studies in fisheries management. The first model is unidimensional, with one control and one state variable. The second model is multidimensional, with one control and two state variables, and includes a diffusion from a source to a sink, thus underscoring the need for a continuous-time framework. Each problem is solved with the three techniques using the same workstation and coding platform, tested in both a PC and MAC environment, with MATLAB R2009 (see the Appendix A on platforms, programs and available programming code). The evaluation of their performance is partly based on the size of approximation errors and computation time, and underscores how the relative superiority of the various techniques changes with respect to the dimension of the problem being solved.

Second, we are not solving a basic classical model to compare techniques, where all qualitative properties are already known. Instead, the numerical case studies in this paper have highly generalised non-linear structures where return functions are dependent on both state and control variables and uncertainty components are state-dependent. These models are increasingly used in applications to biosecurity and fisheries economics, along with other natural resource problems, where not only qualitative properties but more exact numerical results are important to researchers.

Third, as a natural question arising from any exercise that 'picks a winner', we also compare techniques taking into account not only approximation quality and computation time, but other considerations that influence the choice of a technique in practice. For example, all three techniques have different approaches to HJB equations so their software package requirements vary. Another consideration is the fact that each technique may have variants that are more or less efficient to a particular problem. Although it is impossible to report all variants, we nevertheless provide a general guide to the relative advantages and disadvantages of each of these three techniques, their overall ease of use and their scope.

The remainder of the paper is organised as follows. In Section 2, we formulate a generalised dynamic optimisation problem and specify the corresponding HJB equation. The uncertainty components in our generalised formulation are not only driven by Brownian motion but also a Poisson jump-diffusion process, a standard instrument to model randomly discontinuous jumps.[2] Poisson diffusions have been used to model events and generate key results in many studies, for example, in the models of technological progress in Aghion and Howitt (1992) and Walde (1999), interest rate movements in Das (2002) and Piazzesi (2005), and negative shocks to fish stocks in Grafton et al. (2005).

In Section 3, we briefly describe the three techniques to solve the HJB equation for the readers' convenience. The description of the projection and perturbation techniques is especially brief, since they are relatively well-known. Further details for these techniques can be found in Judd (1998). The parametric linear programming technique, on the other hand, is described much more carefully, since it is relatively new. We also provide a theorem, extended from Han and Roy (2009), to accommodate models with Poisson jump-diffusions, thus providing a theoretical basis for this new technique.

Sections 4 and 5 are devoted to the numerical case studies. Each problem is introduced and solved, with reported approximation errors and computation time. With each technique, we solve the problems in the most plain manner, putting aside complicated variants that can be applied to a particular situation. Section 6 addresses the question of 'which technique wins in practice' and Section 7 concludes.

2. A generalised stochastic optimal control problem in a continuous-time setting

We begin with a general optimal control problem in a continuous-time setting. The problem is to identify the maximum value function $V(s)$ and/or the optimal profile of the control variables $c(t)$ such that:

[1] Approaching HJB equations as a partial differential equation and solving via finite difference schemes is often complicated and inconvenient due to the absence of clear boundary conditions (see Hedlund (2003) for a basic illustration of this).

[2] Though involving discontinuous jumps, Poisson processes are considered as continuous diffusions with respect to time as the probability of the discontinuous jump occurring in a time interval converges to zero when the time interval approaches zero.

$$V(s) = \max_{c(t)\in\Phi(k(t))} E_0 \int_0^\infty u(k,c)dt \tag{1}$$

subject to:

$$k(0) = s \tag{2}$$

$$\lim_{\tau\to\infty} \Pr(|k(\tau)| = \infty) = 0 \tag{3}$$

$$dk = g(k,c)dt + \varphi(k,c)dw + \mu(k,c)dq \tag{4}$$

where w is a standard Brownian motion, q is a Poisson jump-diffusion process with an arrival rate $\lambda(k, c)$ and Equation (4) describes the transition of the state variables.

To simplify the notation, we define a functional operator:

$$\begin{aligned} H_c(V(k)) = & -\rho V(k) + u(k,c) + \left(\frac{\partial V}{\partial k}\right)' g(k,c) \\ & + \frac{1}{2} tr\left(\frac{\partial^2 V}{(\partial k)(\partial k)'}\varphi(k,c)'\varphi(k,c)\right) \\ & + \lambda(k,c)[V(k+\mu(k,c)) - V(k)] \end{aligned} \tag{5}$$

so that the HJB equation for problem (1) is:

$$0 = \max_{<c>}\{H_c(V(k))\} \tag{6}$$

This HJB equation can be derived in a heuristic manner by applying the Principle of Optimality introduced by Bellman (1957), as illustrated in Milliaris and Brock (1982). The equation is confirmed in the following theorem:

Theorem 1. *Suppose: (i) k(t) evolves in accordance with* Equations (2)–(4)*; (ii) V(·) is a twice differentiable function which satisfies* Equation (6)*, then V(s) is the maximum value function for problem* (1).

A proof for Theorem 1 (in a comparable setting) is provided in Davis (1993) and Rishel (1990).

3. Parametric approximation approaches to HJB equations

3.1. Projection technique

The projection technique is a natural technique for analytically approximating functional equations and hence applicable to dynamic programming problems (Judd, 1998). In practice, it solves a system of first order conditions and envelope results, and not the HJB equation directly. Specifically, if an interior solution is assumed, we can differentiate the HJB Equation (6) and apply the envelope theorem to obtain a system of differential equations:

$$0 = \frac{\partial H_c(V(k))}{\partial c} \text{ and } 0 = \frac{\partial H_c(V(k))}{\partial k} \tag{7}$$

To approximate Equations (7), the projection technique needs to assume approximate forms for the maximum value and policy functions. These are pre-determined combinations of selected basic functions with undetermined coefficients. The method then tries to determine the coefficients for which the right hand side (RHS) of Equations (7) have the 'smallest distance' to zero. Different definitions of smallest distance lead to different variants of the technique. The most common variant is to find the smallest sum of squares of the RHS of (7) at pre-determined collocation points.

3.2. Perturbation technique

Perturbation theory has a long history in numerical approximation, but its application to dynamic programming is more recent than the projection technique. The perturbation approach to HJB equations was first introduced by Judd and Guu (1993), with a unidimensional problem, and more formally in Gaspar and Judd (1997). The technique starts with adding an auxiliary variable (say ε) into the HJB equation such that if $\varepsilon = 0$ the problem is deterministic, and if $\varepsilon \neq 0$ it is stochastic. Specifically, the HJB equation is converted to:

$$\begin{aligned} 0 = \max_{<c>}\Bigg\{ & -\rho V(k) + u(k,c) + \left(\frac{\partial V}{\partial k}\right)' g(k,c) \\ & + \varepsilon^2 \frac{1}{2} tr\left(\frac{\partial^2 V}{(\partial k)(\partial k)'}\varphi(k,c)'\varphi(k,c)\right) \\ & + \varepsilon\lambda(k,c)[V(k+\mu(k,c)) - V(k)]\Bigg\} \end{aligned} \tag{8}$$

The technique is applied in two rounds. The first round is to solve for the deterministic version by setting $\varepsilon = 0$. After determining the steady state, the method successively differentiates and evaluates the first order conditions and the envelope results of the HJB equation, at the steady state, to solve for the derivatives of the maximum value and policy functions with respect to the state variables. The second round is to differentiate the first order conditions and the envelope results with respect to the auxiliary variable ε and solve for the corresponding derivatives. Once all the necessary derivatives with respect to the state variables and ε are identified, functions of state variables and ε can be constructed using Taylor expansions. The final step is to substitute $\varepsilon = 1$ to obtain the approximation for the stochastic problem.

3.3. Parametric linear programming technique

3.3.1. Theoretical basis of the technique

The linear programming technique approaches the HJB Equation (6) as a system of (weak) inequalities. In this context, the 'max' operator in the equation implies the following two points. First, at any state and any feasible levels of the control variables, the term $H_c(V(k))$ is non-positive. Thus, the maximum value function can be identified from a class of functions which satisfy the system of the weak inequalities imposed by the HJB equation. Second, at a state, there exists at least a feasible control such that the term $H_c(V(k))$ is zero. This is the optimal control for the state in question. Feasible control levels other than optimal ones lead to a strictly negative value. This suggests a way to pin down the optimal policy function. More importantly, it guarantees that the maximum value function is the smallest among those satisfying the HJB (weak) inequalities. This is an important property that we can confirm in the following theorem:

Theorem 2. *Suppose: (i) k(t) evolves in accordance with* Equations (2)–(4)*; (ii) V(s) is the maximum value function for problem* (1)*; (iii) w(s) > 0 for all s, then V(s) uniquely solves the optimization problem*:

$$\min_{J(s)\in C^2} \int w(s)J(s)ds \tag{9}$$

subject to:

$$0 \geq H_c(J(s))$$

for all s and with each s for all $c \in \Phi(s)$.

The proof for Theorem 2 is a basic extension to jump-diffusion processes of the proof contained in Han and Roy (2009). The idea of the theorem is relatively simple. As the maximum value function is the smallest, the sum of its values over a domain with any positive weights is also the smallest. Although the problem is a non-linear minimization problem, it can be approximated with

a linearization process. The detailed technical properties of this approach can be found in Farias and Roy (2004).

3.3.2. Technical procedure for the parametric linear programming technique

There are two key steps in the parametric linear programming technique. First, the maximum value function is conjectured to be a linear combination of some pre-selected basis functions with undetermined coefficients. Suppose the vector of the basis functions is $\Phi(s) = (\Phi_1(s), \Phi_2(s),.. \Phi_K(s))'$ and the vector of the undetermined coefficients is $r = (r_1, r_2,... r_K)'$, where K is the number of coefficients to be estimated, then the maximum value function can be approximated in the form $V(s) = r'\Phi(s)$.

In the second step, the domain of interest is discretized with a set (denoted as G_s hereafter) of 'state collocation points'. Then, with each state collocation point ($s \in G_s$), the associated action correspondence $\Phi(s)$ is discretized with a set (denoted as $G_c(s)$) of 'action collocation points'. These state and action collocation points are used as representatives for the whole state and correspondence spaces. Given the choice of the collocation points the non-linear minimisation problem in Equation (9) can be approximated by the following linear programming scheme:

$$\min_{<r>} \sum_{s\in G_s} w(s) r'\Phi(s) \tag{10}$$

subject to:

$$0 \geq H_c(r'\Phi(s))$$

for all $s \in G_s$ and with each s for all $c \in G_c(s)$.

The size of this linear programming scheme depends on the dimension of the dynamic programming problem and the choice of the collocation points. Denote m as the number of state variables and η as the number of control variables, so that the dimension of the state space is m and the dimension of the action correspondence is η. If we discretize each state variable with κ and each action correspondence with ψ collocation points, the set of state collocation points (G_s) will contain κ^m points. Associated with each point $s \in G_s$, there will be a set $G_c(s)$ containing ψ^η action collocation points. Hence, the constraint system of problem (10) has $\kappa^m \times \psi^\eta$ weak inequalities. As the number of coefficients to be determined is K, the size of the linear programming scheme is $\kappa^m \times \psi^\eta \times K$.

Problem (10) can be solved by any linear programming solver for the undetermined coefficients. Once the maximum value function is approximated, the optimal policy function can be calculated from the HJB equation. In many cases, the structure of the HJB equation allows one to directly solve for the optimal policy function analytically through the first order conditions. In other cases where this is not possible, a grid search for the maximizer over the action correspondence usually works well.

3.3.3. Approximation errors and accuracy improvement

Given the approximation of the maximum value and optimal policy functions, we can calculate the approximation errors following a procedure in Judd (1998). To accomplish this, the two functions are substituted into the HJB Equation (6), and the errors are then calculated as the discrepancies between two sides of the equation, as a percentage of the maximum value function. If the errors are not satisfactory, there are a number of ways to improve the approximation quality. First, the number of state collocation points can be increased. This enhances the representativeness of the state collocation set (G_s) over the domain of interest. However, in this case, the linear programming scheme becomes larger due to the increased number of constraints and the solution is dependent on the capacity of the solver package.

The second measure is to increase the effectiveness of the action collocation sets ($G_c(s)$). A natural but costly way is simply to increase the number of action collocation points. But this again makes the linear programming scheme larger and more costly, computationally. A more efficient way to increase the effectiveness of the action collocation set is to reduce the size of the discretized correspondence. At any state, the constraints in problem (9) bind only at the optimal policy and are slack at all other levels. Thus, if the optimal policy function was known, evaluating the constraints at the optimal level would suffice. Each state would need only one constraint and the number of constraints could be dramatically reduced to κ^m. However, the optimal policy is not known before the approximation process begins and it can even never be perfectly approximated. Nevertheless, if we have some information about the optimal policy levels at any particular state, we can narrow down the relevant correspondence, making the discretization finer, given the same size of action collocation sets. This type of information can come from experience, insights into the problem in question or even a 'draft' solution. In the numerical fishery management problem to follow, for example, we decrease the number of state variables by recognising that the value for the stock of fish may be a 'good distance' away from zero or 'maximum carrying capacity'.

The third measure is to use different weight functions, $w(s)$, as suggested by Han and Roy (2009). A higher relative weight attached to one state will lead to a more accurate approximation at that state. This measure is useful if we need to increase quality in a specific part of the domain. The trade off is that this may reduce the approximation quality in the rest of the domain. Farias and Roy (2004) provide a comprehensive analysis on the constraint sampling property of the parametric linear programming approach in this regard.

Finally, we can enlarge or change the set of basic functions. If more functions are added, there will be more coefficients to be estimated, which results in a higher column dimension of the linear programming scheme. In some situations, choosing another set of basic functions may be helpful. The choice of basic functions can be flexible, at least in many cases. For example, while polynomial functions are the most commonly used, due to their simplicity and convenience, other functions can be chosen if they are thought to be computationally more efficient.

4. A standard unidimensional fishery problem

4.1. The model

In this section, we present a numerical example drawn from fisheries economics. The transition law of a fish population is assumed to have both deterministic and stochastic components. The deterministic component is the difference between a logistic (fish) growth function, with an intrinsic growth rate r, and harvest. The stochastic component consists of two types of diffusions: a Brownian motion and a Poisson process with a negative magnitude. The Brownian process represents 'neutral' natural shocks while the Poisson process represents negative shocks caused, for example, by harvest activities. The magnitudes of both shocks are assumed to be stock dependent.

Denoting s as the fish population, MCC as maximum carrying capacity, r as the intrinsic biological growth rate, h as harvest, w as a standard Brownian diffusion and q as a Poisson diffusion with an arrival rate $\lambda > 0$, it follows that the transition of the fish population is described by a stochastic differential equation given by:

$$ds = \left[rs\left(1 - \frac{s}{\mathrm{MCC}}\right) - h\right]dt + \varphi(s)dw + \mu(s)dq \tag{11}$$

for $\phi(s) > 0$ and $\mu(s) < 0$, the magnitudes of the Brownian and Poisson processes. The profit function for fishing activities is standard. Fishing revenue is $\frac{p}{h^{\alpha}} \times h$ with $0 < \alpha < 1$ where $\frac{p}{h^{\alpha}}$ is the sale price with price elasticity α. Fishing costs per unit are proportional to fish density with a cost parameter c. Fishing profit is thus given by:

$$u(s,h) = ph^{1-\alpha} - \frac{c}{s}h \tag{12}$$

The problem of the regulator is to maximise the aggregate return (discounted at a rate ρ) defined in Equation (12) subject to the transition law defined in Equation (11), given an initial fish stock and the sustainability condition. Specifically, the problem is to approximate the maximum value function $V(s(0))$ such that:

$$V(s(0)) = \max_{h(t),s(t)} \int_0^{\infty} e^{-\rho t} u(s,h)dt \tag{13}$$

subject to Equation (11), $s(0), \lim_{t\to\infty} E_0 s(t) > 0$ and $0 < h(t) \leq s(t) > 0$.

4.2. The HJB equation and numerical values for the parameters

Given the specification for the fishery problem, the HJB equation is:

$$0 = \max_{<h>}\left\{ -\rho V(s) + \left(ph^{1-\alpha} - \frac{c}{s}h\right) + \frac{\partial V}{\partial s}\left[rs\left(1-\frac{s}{\mathrm{MCC}}\right) - h\right] + \frac{1}{2}\frac{\partial^2 V}{\partial s^2}\varphi^2(s) + \lambda[V(s+\mu(s)) - V(s)]\right\} \tag{14}$$

The numerical values for the biological, economic and uncertainty parameters are taken from Grafton et al. (2006), with the price and cost parameters scaled up for graphical convenience. The standard error of the natural shock is assumed to be 5 per cent of the current fish population. Negative shocks which reduce the fish population by 13 per cent are assumed to occur every 10 years. All parameter values are reported in Table 1.

4.3. Technical choice for the approximation process

Given MCC = 1, we approximate the maximum value function in a relatively wide domain [0.2… 1]. The function is conjectured to be a polynomial of order n, $V(s) = \sum_{i=0}^{n} r_i s^i$. There are $n + 1$ parameters to be approximated. We solve the problem for two cases: $n = 6$ and $n = 10$.

The implementation of perturbation technique does not offer much flexibility during its approximation process. However, we have a range of choices for the state collocation points with the projection and linear programming techniques. To make the comparison fair, we discretize the domain into 100 evenly distributed intervals with a set G_s of 101 collocation points and use this for both techniques. For the projection technique, we choose the coefficients to minimise the sum of squares of the RHS of Equations (7). The minimisation algorithm is the quasi-Newton method with a natural initial guess of the coefficients for the iteration process being a zero vector.

Table 1
Numerical values for the parameters in the unidimensional problem.

Name of parameter	Notation	Value
Fish price coefficient	p	700
Fishing cost coefficient	c	17
Price elasticity	α	0.81
Biological intrinsic growth rate	r	0.2985
Discount rate	ρ	0.05
Maximum carrying capacity	MCC	1 (million tons)
Standard error of the natural shock	$\varphi(s)$	0.05 s
Likelihood of the negative shock	λ	0.1
Magnitude of the negative shock	$\mu(s)$	−0.13 s

For the linear programming technique we need action collocation points as well. Hence, for each state s in the collocation set G_s, the action correspondence $[0, s]$ is discretized with a set $G_h(s)$ of 201 nodes. The weight is simply chosen to be the unit vector, $w(s) = 1$ for all s. Given the choice of the state and action collocation points, the constraint system of the linear programming scheme includes $|G_s| = 101$ states. Each state is associated with the $G_h(s) = 201$ constraints corresponding to 201 possible actions in the action collocation set. Thus, the total number of constraints is $|G_s| \times |G_h(s)| = 101 \times 201 = 20,302$. Implementing the algorithms of the three techniques, we have three different analytical approximations of the maximum value functions. Based on these we can calculate the approximation errors.

4.4. Calculating the approximation errors

To calculate the approximation errors, we have to calculate the optimal harvest function first. In this case, the structure of the HJB Equation (14) allows us to calculate the optimal harvest function analytically, or from the first order condition:

$$h = \left[\frac{1}{(1-\alpha)p}\left(\frac{\partial V}{\partial s} + \frac{c}{s}\right)\right]^{-\frac{1}{\alpha}} \tag{15}$$

After the optimal harvest function is calculated from Equation (15), we substitute it into the HJB Equation (14) and divide the RHS by the maximum value function to obtain the approximation errors. The approximated maximum value and optimal harvest functions are plotted in Figs. 1 and 2. The approximation errors of the three techniques are plotted in Fig. 3.

4.5. Performance of the three techniques in the unidimensional case study

To compare the performance of the three techniques, we calculate the average and the maximum of the absolute values of the approximation errors in 101 state collocation points, and report them with the average computation time measured in seconds in Tables 2and 3.

Although actual computation time will vary across individual workstations, the tables convey three clear messages about the performance of the techniques. First, the projection technique is accurate, but slow in comparison to its alternatives. Second, the perturbation technique is the least accurate. This is understandable because the domain of the approximation is wide and perturbation accuracy decays quickly away from the steady state. The third message is the superiority of linear programming technique in this setting. It is not as accurate as the projection technique when $n = 6$, but improves substantially when $n = 10$. The most striking feature is the small extra cost of improving accuracy with respect to computation time, making it faster than the perturbation technique when $n = 10$.

The perturbation technique is renowned for being computational fast, so it is no surprise that it is the best alternative in this regard when $n = 6$. However, the superior speed of the linear programming technique where $n = 10$ is a surprise, needing an explanation. The essential reason lies in the nature of the differences between the algorithms in the two techniques. In spite of the unidimensional system, the perturbation technique has to

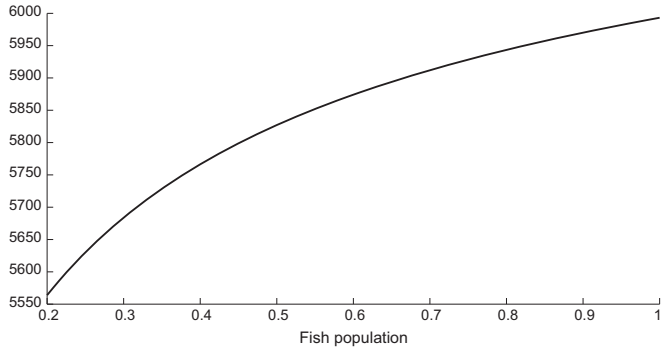

Fig. 1. Unidimensional maximum value function.

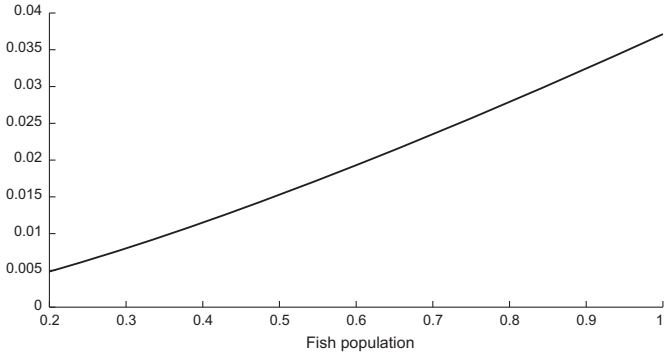

Fig. 2. Unidimensional optimal harvest function.

approximate a two-dimensional polynomial (with respect to the state and the auxiliary variables). This two-dimensional polynomial has 28 coefficients when $n = 6$ and 66 coefficients when $n = 10$. Therefore, an increase in the polynomial order from $n = 6$ to $n = 10$ requires an additional 38 coefficients.[3] With the linear programming technique, the increase in polynomial order involves solving a linear programming scheme with size 11 × 20,302 instead of 7 × 20,302, and only an additional four columns. In other words, the increase in the number of coefficients is quadratic with the perturbation technique while it is linear with the linear programming technique, which makes it faster when $n = 10$.

5. A multidimensional marine reserve problem

5.1. The model

In this section, we solve the marine reserve problem introduced in Grafton et al. (2006). In this model, the authority sets a certain proportion (denoted by R) of a fish population (s) as a reserve. This protected area is closed from harvesting activities. Firms can only catch fish (denoted as h) in the exploitable area (E) with size $1 - R$. The number of fish which transfer from the protected reserve to the exploitable area is proportional to the size of each part and the differential in fish densities, defining a transfer function $T(s_E, s_R)$. The fish stocks in both areas are also subject to a standard Brownian motion w and a Poisson process q with an arrival rate λ. The transition laws are given by:

$$ds_E = \left[rs_E\left(1-\frac{s_E}{(1-R)^*\mathrm{MCC}}\right)+T(s_E,s_R)-h\right]dt+\varphi_E(s_E)dw + \mu_E(s_E)dq \tag{16}$$

$$ds_R = \left[rs_R\left(1-\frac{s_R}{R^*\mathrm{MCC}}\right)-T(s_E,s_R)\right]dt+\varphi_R(s_R)dw+\mu_R(s_R)dq \tag{17}$$

where $T(s_E, s_R) = \phi\frac{R(1-R)}{\mathrm{MCC}}\left(\frac{s_R}{R}-\frac{s_E}{1-R}\right)$. Fishing profit is similar to the standard unidimensional fishery model, except that unit cost is assumed to be proportional to the inverse of the fish density in the exploitable area, or:

$$u(s_E,h) = ph^{1-\alpha}-\frac{c}{\frac{s_E}{(1-R)\mathrm{MCC}}}h \tag{18}$$

The problem of the regulator is to maximise the (discounted) aggregate return defined in Equation (18) subject to the transition laws defined in Equations (16) and (17), given initial fish stocks and the sustainability condition. In particular, the problem is to approximate the maximum value function $V(s_E(0), s_R(0))$ such that:

$$V(s_E(0),s_R(0)) = \max_{h(t),s_E(t)}\int_0^\infty e^{-\rho t}u(s_E,h)dt \tag{19}$$

subject to Equations (16) and (17), $\lim_{t\to\infty} E_0(s_E(t)+s_R(t)) > 0$, given values of $s_E(0)$ and $s_R(0)$, and $0 < h(t) \le s_E > 0$.

5.2. Multidimensional HJB equation and numerical values for the parameters

Given the problem specification, the HJB equation for multidimensional problem is:

$$0 = \max_{<h>}\left\{\begin{array}{l} -\rho V(s_E,s_R)+\left(ph^{1-\alpha}-\dfrac{c}{\frac{s_E}{(1-R)\mathrm{MCC}}}h\right) \\ +\dfrac{\partial V(s_E,s_R)}{\partial s_E}\left[rs_E\left(1-\dfrac{s_E}{(1-R)^*\mathrm{MCC}}\right)+T(s_E,s_R)-h\right] \\ +\dfrac{\partial V(s_E,s_R)}{\partial s_R}\left[rs_R\left(1-\dfrac{s_R}{R^*\mathrm{MCC}}\right)-T(s_E,s_R)\right] \\ +\dfrac{1}{2}\dfrac{\partial^2 V(s_E,s_R)}{\partial s_E^2}\varphi_E^2(s_E)+\dfrac{1}{2}\dfrac{\partial^2 V(s_E,s_R)}{\partial s_R^2}\varphi_R^2(s_R) \\ +\dfrac{\partial^2 V(s_E,s_R)}{\partial s_E\partial s_R}\varphi_E(s_E)\varphi_R(s_R) \\ +\lambda[V(s_E+\mu_E(s_E),s_R+\mu_R(s_R))-V(s_E,s_R)] \end{array}\right\} \tag{20}$$

In a manner similar to the unidimensional case study, the numerical values for the biological and economic parameters are taken from Grafton et al. (2006), as reported in Table 1. The optimal reserve size for this problem is roughly $R = 30\%$, again, as in Grafton et al. (2006).

5.3. Technical choice for the approximation process

At a reserve size $R = 30\%$, the maximum carrying capacity in the protected area is $R^*\mathrm{MCC} = 0.3$ and in the exploitable area is $(1 - R)^*\mathrm{MCC} = 0.7$. We approximate the maximum value function in the domain $(s_E, s_R) \in [0.2\ldots0.7; 0\ldots 0.3]$. The maximum value function is conjectured to be a polynomial of order four,

[3] If we count the coefficients in the optimal policy function to be approximated at the same time, the number will be higher.

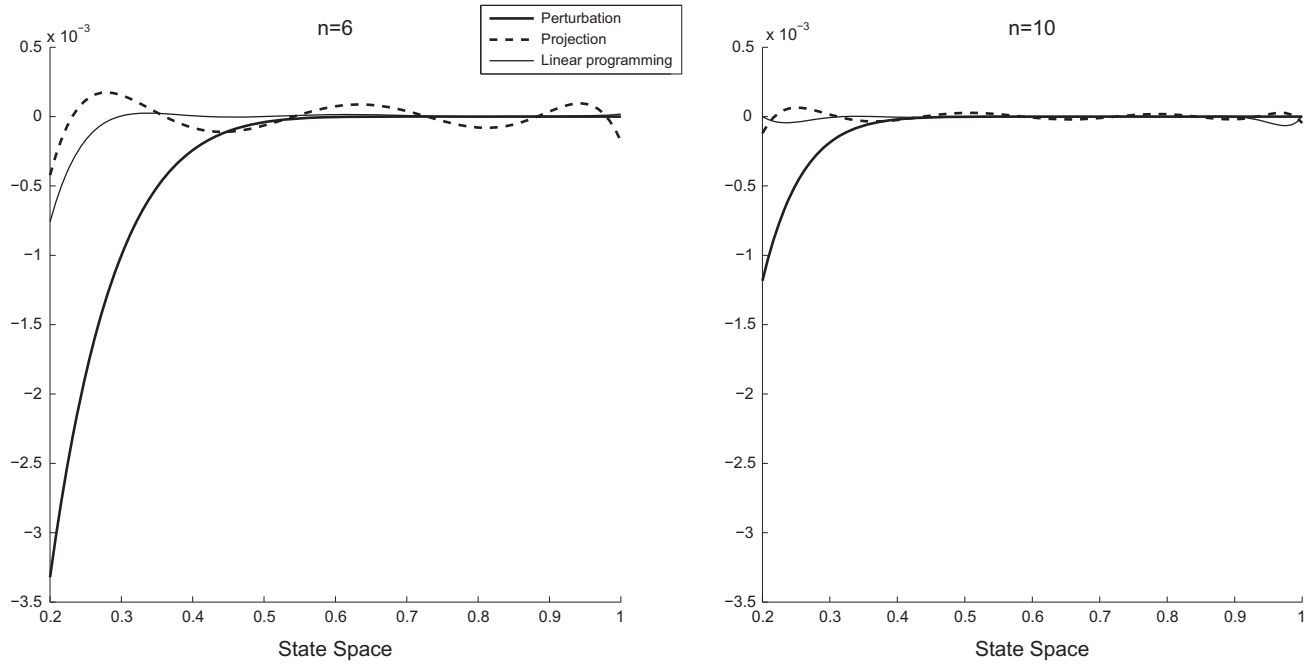

Fig. 3. Unidimensional approximation errors with $n = 6$ and $n = 10$.

$V(s_E, s_R) = \sum_{i=0}^{4}\sum_{j=0}^{4-i} r_{ij}(s_E)^i(s_R)^j$, so that there are 15 coefficients to be approximated.

We discretize the domain and action correspondence into evenly distributed grids. The two-dimensional domain is discretized with a set G_s consisting of $51 \times 51 = 2601$ state collocation points, and we use this for both the projection and linear programming techniques. In a manner similar to the unidimensional case, a quasi-Newton algorithm is used with an initial guess of a zero vector to minimize the residual in the projection method. The linear programming technique needs action collocation points as well. Hence, with each state collocation point $(s_E, s_R) \in G_s$, the action correspondence $[0, s_E]$ is discretized with a set $G_h(s_E, s_R)$ of 51 nodes. The weight is simply chosen to be the unit vector, $w(s_E, s_R) = 1$ for all (s_E, s_R). Given the choice of the state and action collocation points, the constraint system of the linear programming scheme includes $|G_s| = 2601$ states. Each state is associated with $|G_h(s_E, s_R)| = 51$ constraints corresponding to 51 possible actions in the action collocation set. Thus, the total number of constraints is $|G_s| \times |G_h(s_E, s_R)| = 51 \times 51 \times 51 = 132,651$.

5.4. Calculating approximation errors

The structure of the multidimensional HJB Equation (20) again allows us to calculate the optimal harvest function analytically. The first order condition implies:

$$h = \left[\frac{1}{(1-\alpha)p}\left(\frac{\partial V(s_E, s_R)}{\partial s_E} + \frac{c(1-R)}{s_E}\right)\right]^{-\frac{1}{\alpha}} \tag{21}$$

Table 2
Performance of the parametric techniques in the unidimensional problem with polynomial order $n = 6$. Computation time is in seconds.

	Linear programming	Perturbation	Projection
Maximum error	7.5860e-4	3.3214e-3	4.2251e-4
Average error	3.9693e-5	3.4357e-4	7.5792e-5
Computation time (s)	0.2	0.13	0.64

and both the maximum value and optimal harvest functions are substituted into the HJB Equation (20) to calculate the approximation errors. The maximum value, optimal harvest functions and approximation errors produced by the linear programming approach are plotted in Figs. 4–6. Fig. 7 provides example model–output for the time path of optimal harvest and stock values for the exploitable and protected stocks with optimal harvest.

5.5. Performance of the three techniques in the multidimensional case study

As in the unidimensional problem, we calculate the average and the maximum of the absolute values of the approximation errors in the 2601 state collocation points and report them together with the average computation time in Table 4, in seconds. It is clear that both the projection and linear programming techniques are good in terms of quality, while the perturbation technique remains the least accurate. The projection technique, as in the unidimensional case, is again relatively slow.

A notable difference between this and the unidimensional case, however, is that the perturbation technique is now computationally the fastest. The computational time for the linear programming technique is still very good, but relatively slow in comparison. What explains this, of course, is the curse of dimensionality, which generally influences the choice and performance of the techniques in varying ways. A higher dimensional problem requires more coefficients to be approximated in all techniques, implying more derivatives in the perturbation method and more column dimensions in the linear programming method. However, the row dimension of the linear programming scheme has risen significantly in the multidimensional case. In the unidimensional problem, where the state contains 101 collocation points and the correspondence contains 201 points, the row dimension is only $101 \times 201 = 20,302$. In the multidimensional problem, with an additional state variable, and although each state and the action correspondence contains only 51 points, the row dimension is

Table 3
Performance of the parametric techniques in the unidimensional problem with polynomial order $n = 10$. Computation time is in seconds.

	Linear programming	Perturbation	Projection
Maximum error	6.5833e-5	1.1836e-3	1.3641e-4
Average error	9.2673e-6	8.4807e-5	1.9748e-5
Computation time (s)	0.315	1.2	1.47

$51 \times 51 \times 51 = 132{,}651$, or more than six times larger. If we add another state or control variable and discretize it with 51 collocation points, the size of the linear programming problem will be 51 times larger and computation time will increase very quickly.

6. Practical choice over the three techniques

In this section, we discuss some practical considerations on the choice of the three techniques and their applications to dynamic programming problems. Obviously, their performance (e.g., approximation quality and computation time) is an important indicator. However, the choice of the technique in practice also depends on flexibility, applicability, software requirements and problem-specific factors. We take each technique in turn.

6.1. Perturbation technique

The numerical case studies provide evidence that the perturbation technique is not efficient in unidimensional problems, where it remains the least accurate of all three techniques and loses its speed advantage to the linear programming technique. Attempts to improve the accuracy of the perturbation technique, given the number of coefficients to be approximated, are also very limited. The only technique available, thus far, is the use of a rational functional form (or 'Pade approximation') in place of polynomials. However, this can only be applied to deterministic unidimensional HJB equations (see Judd, 1996). In addition, the perturbation technique is the least flexible technique in controlling the distribution of approximation errors, which invariably increase the further away from the steady state (see Fig. 3).

With respect to scope, the perturbation technique is also the most restrictive. As its algorithm relies on differentiating and evaluating the derivatives at a steady state, it works best in problems with a unique steady state. In problems without a steady state (or one that is impossible to calculate), or with multiple steady states, the use of perturbation technique is ruled out. The perturbation technique also cannot work where there exists a corner solution or a non-differentiable HJB equation. In most cases, the perturbation technique also requires an adequate symbolic toolbox, for successive symbolic differentiation and evaluation, which can be relatively expensive software to obtain.

However, the perturbation technique still retains some attractiveness. First, it is only technique that produces a steady state (for deterministic problems) during the approximation process. With the projection and linear programming methods, the calculation of the steady state (if necessary) often needs be done with an additional simulation step, after the approximation of value and optimal policy functions. The perturbation technique is thus convenient in economic problems where the steady state and the dynamic behaviour surrounding the steady state are the main focus of the analysis.

Second, the perturbation technique is still superior for comparative static analysis in terms of changes in parameter values. For example, in the applied cases above, it produces the optimal harvest as a function of both the fish stock and the price of fish, making the analysis of the effect of changes in fish prices on the optimal path to the steady state easy and tractable. This is impossible or extremely difficult with the projection and linear programming techniques.

Finally, the perturbation technique suffers least from the curse of dimensionality, hence it is often the most appropriate in multidimensional problems. In addition, it is the only technique that allows the modeller to increase approximation quality by adding additional terms – a trivial programming exercise compared to the other two techniques. For example, in a unidimensional problem above, in order to improve accuracy by increasing from $n = 6$ to $n = 10$, the perturbation technique does not need to recalculate the coefficients for the first to the sixth order terms. It only has to further calculate the seventh to tenth order terms to generate a longer Taylor expansion. The projection and linear programming techniques do not have this feature as all terms need recalculating when the polynomial order is changed.

6.2. Projection technique

The projection technique is slow, but its accuracy is stable. Practical experience shows that this technique offers the most clear opportunity for a trade off between speed and accuracy in most problems, i.e., the approximation quality can improve significantly

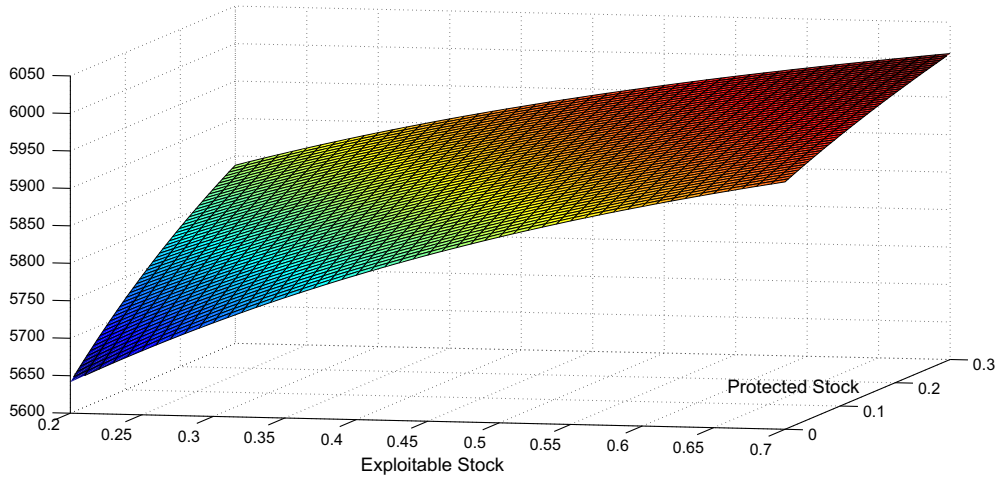

Fig. 4. Multidimensional maximum value function.

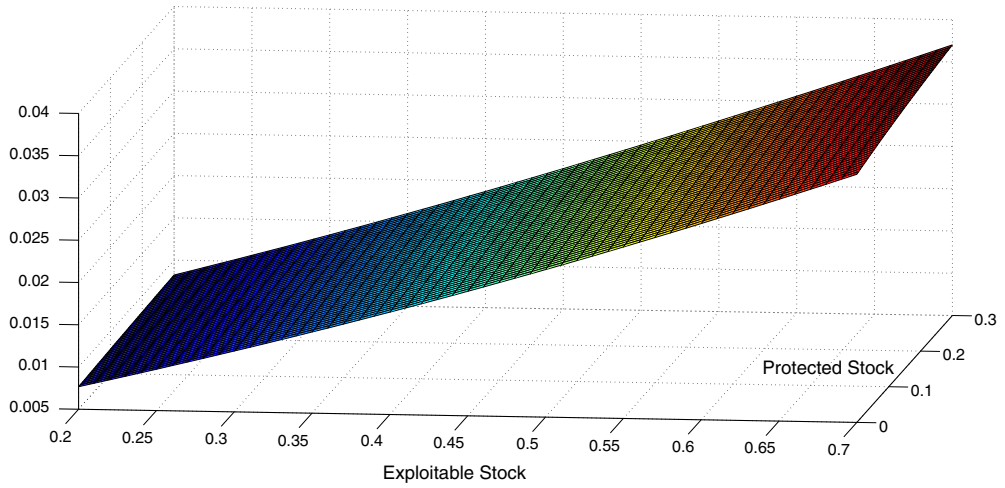

Fig. 5. Multidimensional optimal harvest function.

in most situations when more time is devoted to computation. A higher number of state collocation points or basic functions will lead to significantly more accurate results. In some situations, with the other two techniques, additional computational time leads only to a relatively insignificant improvement in accuracy.

In addition, this technique has a wide range of variants that can be applied flexibly to a particular problem. For example, one may use Chebychev polynomials or other functional forms to improve quality. In short, the variants of the projection technique allow the highest degree of flexibility to control the distribution of the approximation errors. For example, a higher weight attached to a particular collocation point in defining the distance function leads to a smaller error in that state. Even if a researcher needs the errors at some particular points to be zero, the otherwise difficult task of finding coefficients which minimises a relevant distance function can be easily converted to solving a system of equations where the distance is zero in the desired collocation points.

However, due to its slow speed, the projection technique may not be appropriate in many practical situations. For example, economic parameters are usually estimated econometrically, with a confidence interval, requiring often elaborate sensitivity analyses. The projection technique is often too slow in such a setting. Suppose, to illustrate, we calculate the maximum value function of the multidimensional case study with different possible levels of five parameters, namely the price coefficient (p), the cost coefficient (c), the intrinsic rate of growth (r), the reserve size (R) and the likelihood of a negative shock (λ). If the size of the sensitivity analysis is β (each parameter has β possible alternative values), then the total number of parameter combinations is β^5. Based on the computation time for one set of parameter values reported in Table 3, we can calculate the computation time required by the three techniques to implement a sensitivity analysis in Table 5. If the analysis size is $\beta = 5$, the perturbation technique can finish in about a quarter of an hour, the linear programming technique

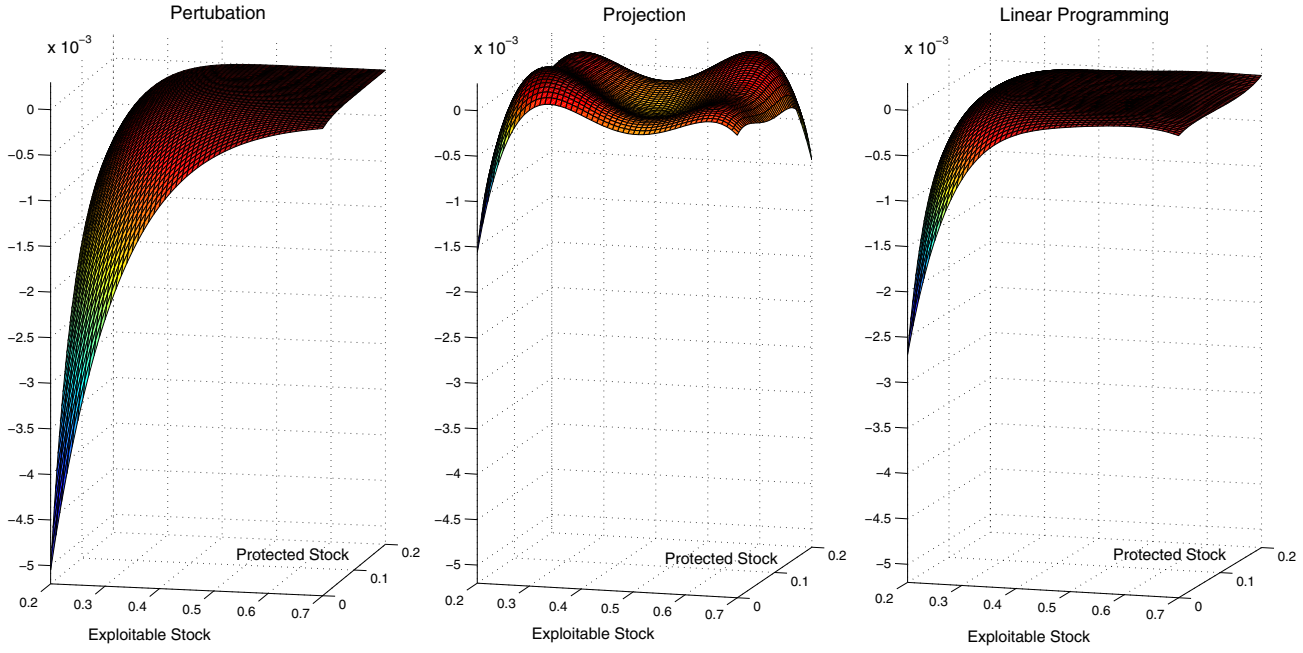

Fig. 6. Multidimensional approximation errors.

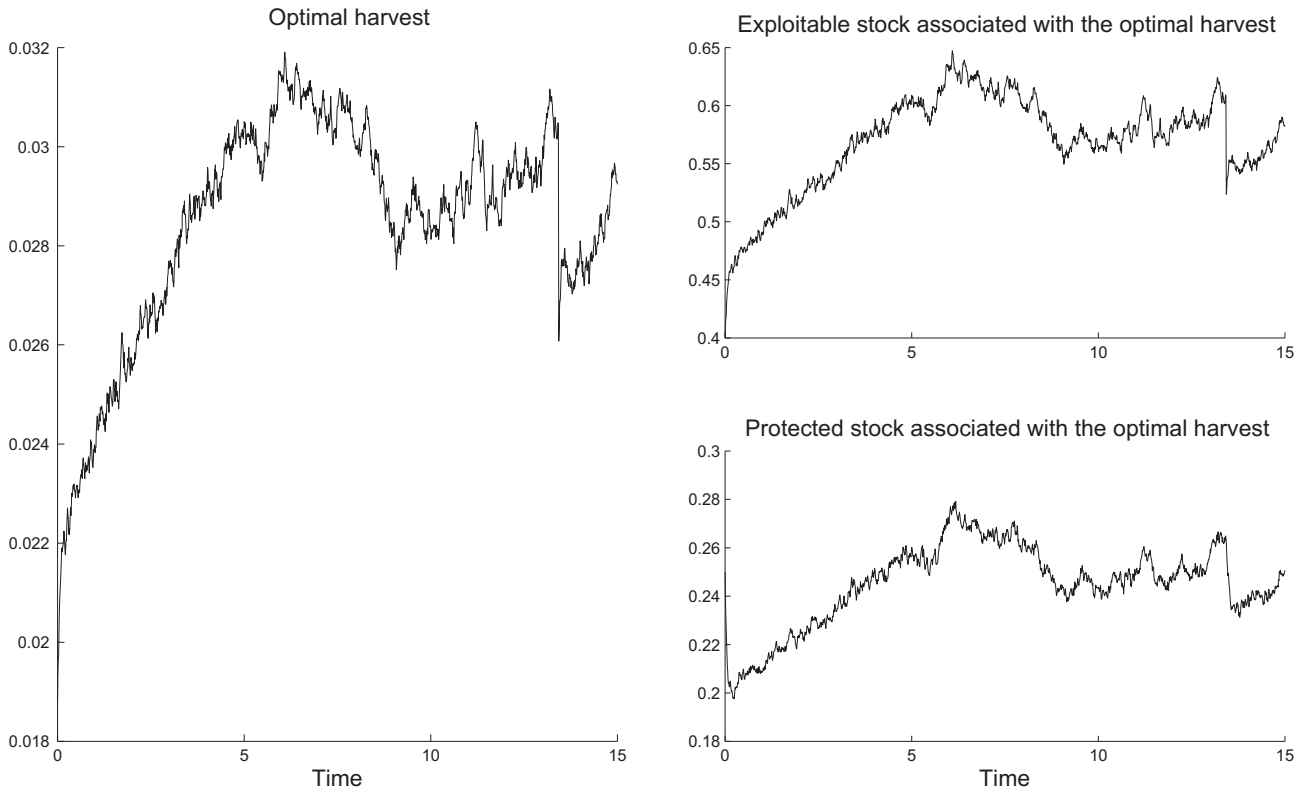

Fig. 7. An example simulated time path for the multidimensional case.

needs about two and a half hours, and the projection technique needs more than 15 h. If the size $\beta = 6$, the perturbation and linear programming techniques can be completed within hours while the projection technique needs more than one and one-half days. In addition, as projection technique uses a non-linear optimisation routine, the computation time will increase very quickly with respect to the number of coefficients. This is especially true when the distance function is very complicated, as the sum of a large number of non-linear terms takes considerable time to be evaluated numerically. Any quality or accuracy gain is quickly offset by otherwise oppressive computational time.

It should also be noted that although applications of the projection technique are much broader than the perturbation technique, since it does not require a steady state, it is still restricted to solving first order conditions and envelope results of HJB equations. Thus, it cannot solve for a corner solution or approximate non-differentiable HJB equations.

6.3. Linear programming technique

The performance of the linear programming technique in the two applied case studies above provides evidence of being both fast and accurate. It strongly competes with the perturbation technique in speed and with the projection technique in approximation quality. Moreover, although the linear programming technique does not have as many possible variants to improve accuracy, compared to the projection technique, the choice of collocation points and functional form can be still be very flexible (as discussed in Section 3). The distribution of approximation errors can also be controlled (somewhat) by choosing different weight-vectors. Perhaps, most importantly, this is the only technique that allows the use of prior information to improve approximation quality. For example, in the unidimensional case study above, an experienced researcher in fisheries economics may 'guess' that the optimal harvest level is certainly less than half of the fish stock. If correct, it is not necessary to discretize the whole action correspondence $[0, s]$ into 201 collocation points. Instead, the researcher can discretize the suspected action correspondence $[0, 0.5s]$ into 101 points and obtain the same result with the row dimension of the linear programming scheme reduced by 50 percent.

In terms of applicability, the linear programming technique offers the widest choices. It relies on neither the existence of a steady state or the need to differentiate HJB equations. It is thus the only technique that can be used in case of a corner solution or with non-differentiable HJB equations.

However, there are some weaknesses. First, the linear programming technique is the only technique that approximates the maximum value function only, so that the optimal policy

Table 4
Performance of the parametric techniques in the multidimensional problem. Computation time in seconds.

	Linear programming	Perturbation	Projection
Maximum error	2.6869e-3	5.10e-3	1.5595e-3
Average error	1.1195e-4	4.1e-4	1.41e-4
Computation time	2.94	0.3	18

Table 5
Computation time and the size of the sensitivity analysis.

Size of the sensitivity (β)	Number of parameter combinations (β^5)	Linear programming technique	Perturbation	Projection
3	243	0:12 h	0:01 h	1:13 h
4	1024	0:50 h	0:05 h	6:07 h
5	3125	2:23 h	0:16 h	15:38 h
6	7776	6:21 h	0:39 h	38:52 h

Table 6
Sizes of the linear programming scheme with various dimensions.

	$\eta = 1$	$\eta = 2$	$\eta = 3$
$m = 1$	2601×5	$132{,}651 \times 5$	$6{,}765{,}201 \times 5$
$m = 2$	$132{,}651 \times 15$	$6{,}765{,}201 \times 15$	$345{,}025{,}251 \times 15$
$m = 3$	$6{,}765{,}201 \times 35$	$345{,}025{,}251 \times 35$	$17{,}596{,}287{,}801 \times 35$

function may not be available in analytical form, as with the two other techniques. When the optimal policy function cannot be solved from the first order conditions of the HJB equation, it has to be approximated numerically, usually from a grid search. It is thus usually costly to obtain the optimal policy function with this method if one needs it to (say) simulate the steady state.

Second, the linear programming problem is most affected by the curse of dimensionality. Adding one variable into the problem, regardless of whether it is a state or control variable, will enlarge the size of the linear programming scheme by several dozen times. Assuming each state or control variable has only 51 collocation points and the value function is conjectured to be a fourth order polynomial, as in the multidimensional case studies, we calculate the sizes of the linear programming schemes with different dimensions and report results in Table 6, where m is the dimension of the state space and η is the dimension of the action correspondence. It is clear that the column size depends on the number of the state variables but the row size increases exponentially with respect to both variables. For the case $m = 3$ and $\eta = 3$, we need at least 4720 GB memory to store the matrix, let alone solve the linear programming scheme. If there are thus more than three state or control variables, the linear programming technique is not practical, unless we accept a highly inaccurate approximation by reducing the number of collocation points to a near trivial level.

Finally, the linear programming technique needs to have a bounded action correspondence. Without boundedness, the action correspondence cannot be discretized into action collocation points. Fortunately, in economics and natural resource management problems, where a decision maker always faces scarcity constraints, most optimal control problems already satisfy, or can be converted to satisfy, the boundedness requirement easily. But in other contexts, this will not be the case.

7. Closing remarks

In this paper, we extend the parametric linear programming approach to include problems with jump-diffusion processes and compare it to the projection and perturbation techniques. The performance of the three techniques is illustrated in two case studies in fisheries management, and varies considerably in terms of computation time, accuracy, scope and ease of use.

Results will always depend on the specific workstation used, and the nature of the problem being solved, but the case studies used in this paper do provide a set of general guidelines. First, the linear programming technique provides a nice combination of speed and accuracy, with basic software requirements, and is widely applicable to cases where HJB equations are not even differentiable. However, the linear programming technique is most affected by the curse of dimensionality, making it unsuitable for large dimensional problems, at least with current desktop computing technology.

Second, in this respect the perturbation technique is still an attractive alternative since is can handle large scale problems with reasonable computation time, making it ideal for sensitivity analysis on parameter values. It's weakness is its need for a unique steady state and its lack of accuracy, which decays rapidly away from a steady state value.

Finally, the projection technique is relatively slow, especially in multidimensional problems. However, it has a wide range of variants that may be applied to particular settings and models. It is also fairly intuitive, easy to code and requires only a non-linear optimiser which is widely available in many coding platforms.

Acknowledgements

Helpful comments from three referees and funding from the Commonwealth Environment Research Facilities program from the Australian Department of Sustainability, Environment, Water, Population and Communities is gratefully acknowledged.

Appendix A. Platform, programs and program code

The two cases studies are solved with the three techniques using the same workstation and coding platform. The workstation has 16 GB RAM with QuadCore i5 Intel processors. It runs in MAC environment with OSX 10.6 operating system, and in Windows 7.

All software required to run the code is available on the corresponding authors webpage. The coding platform is MATLAB R2009b. The symbolic manipulation required in perturbation method (successively differentiating the first order conditions and the envelope results) is done via the Maple Toolbox for Matlab 15. Linear programming schemes are solved with Tomlab, a Matlab add-on package. Residual minimization required in projection technique is implemented via Quasi-Newton method, a built-in function in the coding platform.

References

Aghion, P., Howitt, P., 1992. A model of growth through creative destruction. Econometrica 60 (2), 323–351.
Arruda, E., DoVal, J., 2006. Approximate dynamic programming based on expansive projections. In: Proceedings on 45th IEEE Conference on Decision and Control, 5537–5542 pp..
Aruoba, S., Jesus, F., Juan, F., 2006. Comparing solution methods for dynamic equilibrium economies. Journal of Economic Dynamics and Control 30 (12), 2477–2508.
Bellman, R., 1957. Dynamic Programming. Princeton University Press, New Jersey.
Clark, C.W., 1976. A delayed-recruitment model of population dynamics, with an application to baleen whale populations. Journal of Mathematical Biology 3 (3–4), 381–391.
Das, S., 2002. The surprise element: jumps in interest rates. Journal of Econometrics 106 (1), 27–65.
Davis, M.H.A., 1993. Markov Models of Optimization. Chapman and Hall, London.
Doraszelski, U., Judd, K., 2010. Avoiding the Curse of Dimensionality in Dynamic Stochastic Games. Manuscript.
Farias, D., Roy, B., 2004. On constraint sampling for the linear programming approach to approximate dynamic programming. Mathematics of Operations Research 29 (3), 462–478.
Gaspar, J., Judd, K., 1997. Solving large-scale rational-expectations models. Macroeconomic Dynamics 1 (1), 45–75.
Grafton, Q., Kompas, T., Lindemayer, D., 2005. Marine reserves with ecological uncertainty. Bulletin of Mathematical Biology 67, 957–971.
Grafton, Q., Kompas, T., Pham, H., 2006. The economic payoffs from marine reserves: resource rents in a stochastic environment. Economic Record 82, 469–480.
Han, J., Roy, B., 2009. Control of diffusions via linear programming. In: Infanger, G. (Ed.), Stochastic Dynamic Programming: The State of the Art. Manuscript in Honor of George B. Dantzig.
Hedlund, S., 2003. Computational Methods for Optimal Control of Hybrid Systems. PhD thesis, Department of Automatic Control, Lund Institute of Technology, Sweden.
Judd, K., 1996. Approximation, perturbation, and projection methods in economic analysis. In: Amman, H., Kendrick, D.A., Rust, J. (Eds.), 1996. Handbook of Computational Economics, vol. 1. Elsevier, London.
Judd, K., 1998. Numerical Methods in Economics. The MIT Press, Cambridge.
Judd, K., Guu, S., 1993. Perturbation solution methods for economic growth model. In: Varian, H. (Ed.), Economic and Financial Modeling with Mathematica. Springer Verlag, New York.
Manne, A., 1960. Linear programming and sequential decisions. Management Science 6 (3), 259–267.
Milliaris, A.G., Brock, W.A., 1982. Stochastic Methods in Economics and Finance. Elsevier Science, Amsterdam.
Nicol, S., Chades, I., 2010. Beyond stochastic dynamic programming: a heuristic sampling method for optimizing conservation decisions in very large state

spaces. Methods in Ecology and Evolution. doi:10.1111/j.2041-210X.2010.00069.x.
Piazzesi, M., 2005. Bond yields and the federal reserve. Journal of Political Economy 113 (2), 311–344.
Rishel, R., 1990. Controlled continuous-time Markov processes. In: Heyman, D.P., Sobel, M.J. (Eds.), 1990. Handbook in Operations Research and Management Science, vol. 2. North-Holland, Amsterdam.
Ross, S., 1970. Applied Probability Models with Optimization Applications. Holden-Day, San Francisco.
Taylor, J., Uhlig, H., 1990. Solving nonlinear stochastic growth models: a comparison of alternative solution methods. Journal of Business and Economic Statistics 8 (1), 1–17.
Walde, K., 1999. Optimal saving under poisson uncertainty. Journal of Economic Theory 87 (2), 194–217.