Solving Higher-dimensional Continuous Time Stochastic Control Problems by Value Function Regression*

Michael Reiter

March 1997, revised June 1998

Adress of the author:

Michael Reiter
Universitat Pompeu Fabra – Department of Economics
Ramon Trias Fargas, 25–27
08005 Barcelona
Spain

e-mail: reiter@upf.es

*I am grateful to Kenn Judd, Albert Marcet, Harald Uhlig and an anonymous referee for very helpful discussions and suggestions, and to Anargyros Papageorgiou and Joseph Traub for providing the FINDER software to generate low discrepancy sequences.

Abstract

The paper develops a method to solve higher-dimensional stochastic control problems in continuous time. A finite difference type approximation scheme is used on a coarse grid of low discrepancy points, while the value function at intermediate points is obtained by regression. The stability properties of the method are discussed, and applications are given to test problems of up to 10 dimensions. Accurate solutions to these problems can be obtained on a personal computer.

Key words: Dynamic Programming, stochastic control, approximation
JEL classification: C 61

1 Introduction

Applications of dynamic programming for models with state space of dimension higher than two or three are plagued by the curse of dimensionality: the complexity of the problem grows exponentially with the dimension of the problem. Specifically, the number of grid points on which the solution is computed usually has to grow exponentially with the number of dimensions in order to get a certain level of accuracy.

This result holds on a worst case basis (Rust 1996). One might hope that for average, well-behaved problems one can find methods that are still practical in higher dimensions. This is of particular relevance in economics, where most problems use smooth functional forms and have smooth solutions. The well known linear-quadratic approximations already provide solutions that are sufficiently accurate for many purposes. Higher order approximations can be obtained by perturbation methods (Judd 1996). Disadvantages of both methods are that they are based only on a local approximation about a steady state, and that restrictions on the control vector cannot easily be imposed. More accurate global approximations can be obtained by projection methods with polynomial bases. Judd (1992) has argued that these are suitable for the treatment of higher-dimensional problems. In one- or two-dimensional problems, projection methods have been shown to give very exact solutions in a very short time. In higher dimensions, however, these methods face the problem that the number of base functions n grows rather rapidly with the number of dimensions, and that an n-dimensional nonlinear minimization problem must be solved (Rust 1996).

The latter problem was one of the motivations to explore a different route. In this paper, I will try to exploit the smoothness of dynamic economic problems, yet stay within the dynamic programming approach, based on value function and policy function iterations on a finite grid. No high-dimensional nonlinear maximization or root finding problems are encountered here. I will use relatively small grids (about 1000-8000 points) for problems with up to 10 dimensions, and approximate the value function locally, based on nearby points in the grid. I should stress that the paper makes no claim that this method succeeds in breaking the curse-of-dimensionality. As a referee has pointed out, the approximation scheme used here faces a curse of dimensionality problem, at least on a worst case basis. What the paper rather tries to show is that many nonlinear models

of moderate dimension (up to about ten state variables) can be accurately solved on a PC. I have worked out the method for continuous time problems; similar ideas may be applied to discrete time problems.

Standard numerical applications of dynamic programming use a uniform grid. In higher dimensions, this is not really an option since the number of grid points "per dimension" is very small: in the biggest 10-dimensional example to follow, it is $8000^{1/10} = 2.46$, which is not easily implemented with a uniform grid. A more flexible approach uses so-called subrandom sequences or low discrepancy points. These points have two desirable properties: They fill the space relatively uniformly, thereby avoiding the loss of information that comes from points which are too close to each other. Second, it is possible to use an arbitrary number of points, regardless of the dimension. Examples of low discrepancy points are Halton, Sobol and Faure points. Sobol points performed best in the Monte-Carlo integration study of (Paskov 1994), and they are implemented in Press, Flannery, Teukolsky and Vetterling (1986). Later studies (Papageorgiou and Traub 1996) indicate that generalized Faure points perform even better in financial applications. The applications in the present paper use Sobol and generalized Faure points. The use of low discrepancy points in the context of dynamic programming was advocated by Rust (1996). Rust (1997b) provides the theoretical result that methods based on random sequences break the curse of dimensionality in a stochastic sense. Rust (1997a) has shown that policy iteration on low discrepancy grids outperforms methods based on either random grids or uniform grids for smooth problems even with only one or two dimensions.

I will refer to the method that I will present as "Value Function Regression" method, abbreviated VFR, since the value function will be approximated by regression methods. The idea of value function approximation has been used before (cf. for example Keane and Wolpin, 1994). Contrary to their application, I consider infinite horizon problems and the approximation is done implicitly in every policy iteration step.

The plan of the paper is as follows: In Section 2 I will describe the general form of the problems to be solved, and describe briefly the relevant aspects of standard methods to solve continuous time stochastic control problems. Section 3 describes the Value Function Regression method, discusses its main properties, convergence issues and some

computational aspects. Test applications are given in Section 4. Section 5 conludes.

2 Continuous time stochastic control problems

2.1 The general model

The problems considered in this paper are of the form

$$\min_{\alpha} \mathrm{E}\left[\int_0^{\infty} e^{-\rho t} u(x(t), \alpha(t))\, dt\right] \tag{1}$$

subject to

$$dx = f(x, \alpha)dt + \sigma(x)\, dz \tag{2}$$

where x is a vector of state variables, α a vector of control variables, and dz a vector of independent Brownian motion processes. To simplify the discussion, the following formulas refer to the case where the matrix σ is diagonal. Section 3.4 describes how to deal with the general case in the framework of the VFR method. I will assume throughout that σ only depends on x, but it would not be difficult to allow σ to depend on the control α.

2.2 Locally consistent approximations

This section briefly summarizes those aspects of the standard methods of stochastic control that are immediately relevant for the method I will develope in Section 3. Continuous time stochastic control problems are treated by a discretization technique based on locally consistent approximations. The general framework is developed in Kushner and Dupuis (1992).

For the discretization we choose a time unit Δt and a discrete grid, where h_i denotes the distance between grid points in the i-direction. Since we deal with diffusion processes, from time t to $t + \Delta t$ the system can only reach neighbouring points, which differ from the current point in only one dimension. We define $p^h(x, x \pm e_i h_i | \alpha)$ as the probability that the system will reach the point $x \pm e_i h_i$ at time $t + \Delta t$, conditional on being at x at time t, and for a given control vector α. Here, e_i is a vector with zero elements, except

of element i which is equal to 1. Transition probabilities are determined by

$$p^h(x, x \pm e_i h_i|\alpha) = \left(\frac{\sigma_i^2/2}{h_i^2} + \frac{f_i^{\pm}(x,\alpha)}{h_i}\right)\Delta t^h \tag{3}$$

and

$$p^h(x, x|\alpha) = 1 - \sum_i \Big(p^h(x, x + e_i h_i|\alpha) + p^h(x, x - e_i h_i|\alpha)\Big)$$

where f^+ and f^- denote the positive and the negative part of function f, respectively. To make sure that all probabilities lie between 0 and 1, it is necessary to choose Δt^h small enough, for example by

$$\Delta t^h = 1/\max_x \sum_i \left(\frac{\sigma_i^2}{h_i^2} + \frac{f_i(x,\alpha)}{h_i}\right) \tag{4}$$

Using (3) we obtain (arguments were dropped for notational simplicity)

$$\mathrm{E}\,\Delta x_i = \left[\left(\frac{\sigma_i^2/2}{h_i^2} + \frac{f_i^+}{h_i}\right) - \left(\frac{\sigma_i^2/2}{h_i^2} + \frac{f_i^-}{h_i}\right)\right] h_i \Delta t^h = f_i \Delta t^h \tag{5a}$$

$$\begin{aligned}\mathrm{E}\,\Delta x_i^2 &= \left[\left(\frac{\sigma_i^2/2}{h_i^2} + \frac{f_i^+}{h_i}\right) + \left(\frac{\sigma_i^2/2}{h_i^2} + \frac{f_i^-}{h_i}\right)\right] h_i^2 \Delta t^h \\ &= \sigma_i^2 \Delta t^h + |f_i| h_i \Delta t^h = \sigma_i^2 \Delta t^h + O(h_i^3)\end{aligned} \tag{5b}$$

$$\mathrm{E}\,\Delta x_i \Delta x_j = 0 \tag{5c}$$

As equations (5) show, the discretization gives the right values for the first two moments of the change in x and is therefore called "locally consistent". Based on local consistency, Kushner and Dupuis (1992) proove that the solution of the discretized system converges to that of the continuous time system.

3 The Value Function Regression method

In this section I develop a method to solve higher-dimensional stochastic control problems in continuous time. The idea is to compute the value function on a relatively coarse grid of low discrepancy points, and find values at different points by regression.

3.1 Description of the method

For a problem with d state variables, we choose a discrete time step Δt and a set of N grid points $\mathcal{X} = \{x^n\}_{n=1,\dots,N}$ where $x^n \in \mathcal{R}^d$. If, at time t, the dynamic system is

in state x^n, it will either stay there at time $t + \Delta t$ or change to a neighbouring state $x^n + h_j e_j, \quad j = 1, \ldots, d$. Since the grid $\mathcal{X}$ is very coarse, such a point will not be an element of $\mathcal{X}$. Let us denote the set of $(2d \cdot N)$ neighbouring points by $\mathcal{X}^\circ$.

We now come to the task of computing the value function $V(x;\alpha)$ for a given set of controls α. Collect the values $V(x;\alpha)$ for $x \in \mathcal{X}$ in the vector $v(\alpha)$, those for $x \in \mathcal{X}^\circ$ in the vector $v^\circ(\alpha)$. The vector $u(\alpha)$ collects the values of instantaneous utility during the time period Δt, $u(x,\alpha) \cdot \Delta t$, for $x \in \mathcal{X}$. These vectors satisfy the equation

$$v(\alpha) = u(\alpha) + e^{-\rho \Delta t} \begin{bmatrix} \Pi_{11}(\alpha) & \Pi_{12}(\alpha) \end{bmatrix} \begin{bmatrix} v(\alpha) \\ v^\circ(\alpha) \end{bmatrix} \tag{6}$$

where the matrices $\Pi_{11}(\alpha)$ and $\Pi_{12}(\alpha)$ are the appropriate parts of the usual transition probability matrices. Of course, utility function and transition probability matrices depend on the control α.

Since $\mathcal{X}^\circ$ is not part of the grid, the vector v° must be estimated from v by some form of approximation. If we choose a linear approximation scheme, we can write $v^\circ = Qv$ with an $N \times (2d \cdot N)$ matrix Q. Then (6) becomes

$$v(\alpha) \approx u(\alpha) + e^{-\rho \Delta t} \left[\Pi_{11}(\alpha) + \Pi_{12} Q(\alpha) \right] v(\alpha) \tag{7}$$

$$= u(\alpha) + e^{-\rho \Delta t} \Pi^*(\alpha) v(\alpha) \tag{8}$$

where $\Pi^*(\alpha) = [\Pi_{11}(\alpha) + \Pi_{12} Q(\alpha)]$. Equation (8) looks very similar to a standard value equation. There is one important difference, however: In the standard case, the matrix $\Pi(\alpha)$ is a stochastic matrix, i.e., it has non-negative elements and the elements in a row add up to unity. This guarantees that $v(\alpha) = u(\alpha) + e^{-\rho \Delta t} \Pi(\alpha) v(\alpha)$ has a unique solution, to which the value iteration $v_{k+1}(\alpha) = u(\alpha) + e^{-\rho \Delta t} \Pi(\alpha) v(\alpha)$ converges. In addition, the policy iteration scheme $v(\alpha_{k+1}) = \max_\alpha u(\alpha) + e^{-\rho \Delta t} \Pi(\alpha) v_k(\alpha_k)$ globally converges to the solution of the Hamilton-Jacobi-Bellman equation $v = \max_\alpha u(\alpha) + e^{-\rho \Delta t} \Pi(\alpha) v(\alpha)$. The matrix $\Pi^*(\alpha)$, on the other hand, also contains *negative* elements. Usual convergence proofs do therefore not apply to this method, and the stability of the algorithm is our primary concern. Before investigating the issue of convergence, I will be more specific on how to compute the approximation matrix Q in the next subsections.

3.2 Approximation by regression

There are many possible choices for the matrix Q. The method I use is a linear least squares regression, where the regression at a given point is local in the sense that it is only based on nearby points. The local approximation has two advantages. First, it is more flexible than a global approximation and does not impose a functional form on the value function. This can be used in a next step to investigate the obtained value function and improve the regressors. Second, it leads to a matrix Π^* that is relatively sparse, allowing the use of more grid points. In the following, I describe the approximation scheme which proved to be the most useful in the applications. The value of $v\left(x+h_i e_i\right)$ is approximated by

$$v\left(x+h_i e_i\right) \approx v(x) + \frac{\partial v(x)}{\partial x_i} h_i + \frac{1}{2}\frac{\partial^2 v(x)}{\partial x_i^2} h_i^2, \qquad i = 1, \ldots, d \tag{9}$$

Estimates of the partial derivatives are obtained by regression, fitting a linear combination of M known functions q_m:

$$v(x) = \sum_{m=1}^{M} \beta_m q_m(x) + \epsilon \tag{10}$$

To present the details of the approximation, let us define Z, DZ_j and D^2Z_{jk} as $N \times M$ matrices of which element (n,m) is given by $q_m(x^n)$, $\frac{\partial q_m}{\partial x_j}(x^n)$ and $\frac{\partial^2 q_m}{\partial x_j x_k}(x^n)$, respectively. Since, in the neighborhood of point x^n, we base the approximation (10) only on the N_a points that are closest to x^n, we extract the components belonging to those points: Let Z^n denote the $N_a \times M$ matrix which collects the rows of Z belonging to the N_a nearest points of x^n, and define DZ_j^n and $D^2Z_{jk}^n$ accordingly. The vector v^n gives the value function at these points. In the neighborhood of x^n, we then apply (10) by using β^n, the weighted least-squares estimate of β obtained by

$$\hat{\beta}^n = \left(Z^{n\prime}\Omega^n Z^n\right)^{-1} Z^{n\prime}\Omega^n v^n \tag{11}$$

The weighting matrix Ω^n reflects the different weights of different observations. Since the approximation is local, the idea is to give more weights to nearby points. The first and second derivatives of $v(x)$ w.r.t. x_j and x_k are then given by $DZ_j^n\hat{\beta}^n$ and $DZ_{jk}^n\hat{\beta}^n$, and approximation (9) becomes

$$v\left(x^n+h_i e_i\right) \approx v(x^n) + DZ_i^n\hat{\beta}^n h_i + \frac{1}{2}DZ_{ii}^n\hat{\beta}^n h_i^2 \tag{12}$$

Equations (12) and (11) provide an approximation for $(x^n + h_i e_i)$ that is linear in v and therefore determine the matrix Q in (7). This construction guarantees that the row sums of Q are equal to 1 if a constant is among the approximating functions q_j. This then guarantees that the row sums of $\Pi^*(\alpha)$ are equal to 1.

3.3 Choice of approximating functions

It remains to find useful functions q_m. The most obvious candidates are polynomials in the state variables or in some transformation of the state variables. Second-order polynomials were therefore always included in the matrix Z^n. Using higher order approximations is problematic, however: just as in econometrics, too many regressors, compared to the number of data points (here: compared to the number of nearby points N_a), will yield inaccurate forecasts. Rather than including higher order polynomials right from the start, I use the following iterative strategy: I start using only quadratic functions on x or a transformation of x that seems plausible given the functional form of cost function and dynamic equation. Since the approximation is only locally quadratic, the estimate of the value function so obtained is not quadratic and can be used to find better regressors. First, from some set of nonlinear transformations $f(x)$ (power transformations or Box-Cox transformations are natural candidates, cf. Section 4.3) I choose f such that the value function can best be approximated, in terms of R^2, as a quadratic function in $f(x)$. Then I check the cubic functions in $f(x)$ to see which ones are most useful in explaining the value function, and include a small number (about one half of the number of quadratic functions, $d(d+1)/4$) of them in the regressor set. Then I choose the number of nearby points N_a about three times as big as the number of regressors.

The above procedure is obviously rather simple and ad hoc. The approximation could normally be improved by incorporating any available information on the solution. One could use, for example, perturbation or projection methods (Judd 1996), or methods that give accurate approximations to the deterministic version of the model, to generate suitable base functions (this is in the spirit of what Judd (1996) calls "hybrid methods"). Since the search for better approximating functions is specific to any particular application, I have confined myself here to the above simple strategy.

3.4 The choice of neighbouring points

Since the neighbouring points are not part of the grid, the step sizes and directions can be chosen independently of the grid, and can be different at each point of the grid. A good choice for the step size is

$$h_i = 0.5 \left[|b_i| \cdot d \cdot \Delta t + \sqrt{(|b_i| \cdot d \cdot \Delta t)^2 + 4\sigma_i^2 d \cdot \Delta t} \right] \tag{13}$$

so that the system reaches every neighbouring point with probability $1/(2d)$. This means that the dynamic system has the desirable property to "mix" fast (Kushner and Dupuis 1992, p.100).

The freedom to choose the neighbouring points has three advantages. First, and most importantly, we can choose h and Δt small enough so that the local consistency conditions (5) are satisfied rather accurately. (The optimal choice of h is determined by roundoff error considerations.) Second, we can easily allow for non-diagonal σ in (2), that means non-diagonal covariance matrices. Rather than the unit vectors e_i, we choose step directions conforming to the eigenvectors of the covariance matrix $\sigma\sigma'$. The quadratic approximations (9) and (12) are modified in the obvious way, but otherwise the nondiagonality causes no problem. This appears to be simpler than the modifications necessary in the standard method to account for strong correlation among error terms (Kushner and Dupuis 1992, Sect. 5.3.1). Third, small h facilitates the computation of optimal controls in each policy iteration. This is explained in the next subsection.

3.5 Optimal controls and first order conditions

In each policy iteration step, optimal controls have to be found based on the value function $V(x)$ that was obtained in the preceding step. Computation time is greatly reduced by the fact that we need not solve the exact problem of the discretized model, but can take the solution of the first order conditions of the continuous time model, if h is chosen small enough. To show this, note that the optimal control of the discretized model at point x is given by

$$\begin{aligned} \min_{\alpha} \; & u(\alpha)\Delta t^h + e^{-\rho\Delta t}\Big[p^h(x,x|\alpha)V^h(x) \\ & + \sum_i \Big(p^h(x,x+e_ih_i|\alpha)V^h(x+e_ih_i|\alpha) + p^h(x,x-e_ih_i|\alpha)V^h(x-e_ih_i|\alpha)\Big)\Big] \end{aligned} \tag{14}$$

and that the minimand can be approximated as

$$
\begin{aligned}
& u(\alpha)\Delta t^h + \\
& \qquad e^{-\rho\Delta t}\left[p^h(x,x|\alpha) + \sum_i \left(p^h(x,x+e_ih_i|\alpha) + p^h(x,x-e_ih_i|\alpha)\right)\right] V^h(x) \\
& \qquad\qquad + e^{-\rho\Delta t}\sum_i \left(p^h(x,x+e_ih_i|\alpha) - p^h(x,x-e_ih_i|\alpha)\right) V_i^h(x)h_i \\
& \qquad + \frac{1}{2}e^{-\rho\Delta t}\sum_i \left(p^h(x,x+e_ih_i|\alpha) + p^h(x,x-e_ih_i|\alpha)\right) V_{ii}^h(x)h_i^2 + O(h^3) \\
& = u(\alpha)\Delta t^h + e^{-\rho\Delta t}\left[V^h(x) + \sum_i f_iV_i^h(x)\Delta t^h + \frac{1}{2}\sum_i \left(\sigma_i^2\Delta t^h + |f_i|h_i\Delta t^h\right) V_{ii}^h(x)\right] \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad + O(h^3)
\end{aligned}
$$

using the local consistency conditions (5a) and (5b). Dividing by Δt^h and omitting constants we get the equivalent problem

$$\min_\alpha u(\alpha) + \sum_i f_iV_i^h(x) + \frac{1}{2}\sum_i \left(\sigma_i^2 + |f_i|h_i\right) V_i^h(x) + O(h) \tag{15}$$

Denoting by α^* the solution to the first order condition

$$\frac{du(\alpha^*)}{d\alpha} = -\sum_i \frac{df_i(\alpha^*)}{d\alpha}V_i$$

and by α^{min} the solution to (15), we see that

$$\alpha^{min} = \alpha^* + O(h)$$

provided continuity of the relevant functions. For h small enough, we can therefore replace α^{min} by α^*.

3.6 Summary of the algorithm

1. Fix N, the number of grid points, N_a, the number of nearby points used for regression, and the approximating functions $q_m, \quad m = 1, \ldots, M$.

2. For any point $x \in \mathcal{X}$ find the N_a closest points.

3. Compute the matrix Q so that $v^\circ = Qv$, by the methods described above.

4. Determine starting values for the controls α^0, for example by a linear-quadratic approximation.

5. Find the vector $v^k(\alpha^k)$ by solving equation (8).

6. Find the controls α^{k+1} by solving

$$\max_{\alpha} u(x,\alpha) + \sum_{i=1}^{d} f^i(x,\alpha)\frac{\partial v(x;\alpha)}{\partial x_i}$$

for each x in $\mathcal{X}$. The partial derivatives are obtained from the regression procedure.

7. Repeat steps (5) and (6) until convergence is achieved (in practice: until $\left|\left|v^{k+1} - v^k\right|\right|_{sup} < 10^{-6}\left|\left|v^k\right|\right|_{sup}$).

8. Use the obtained solution to find better regressors, as explained in Section 3.3. Redo steps (5) – (7). Eventually repeat several times.

Some details of the numerical implementation will be discussed in Section 3.10.

3.7 Convergence

In the standard dynamic programming approach, the convergence of the policy iteration scheme can be proven under relatively weak conditions (Kushner and Dupuis 1992, p.155). As already mentioned, this is not the case for the VFR method, since the analogue of the transition matrix is not a stochastic matrix here. The following paragraphs go through the steps of a standard proof, to see where things go wrong, and to see under what conditions we can nevertheless expect to get convergence of the VFR method.

If we abbreviate the matrix $e^{-\rho\Delta t}\Pi^*(\alpha)$ by $R(\alpha)$, we have, by definition of $v(\alpha_k)$, that

$$v(\alpha_k) = R(\alpha_k)v(\alpha_k) + u(\alpha_k)$$

for each policy iteration k. Iterative substitution leads to

$$v(\alpha_k) = R^m(\alpha_k)v(\alpha_k) + \sum_{i=0}^{m-1} R^i(\alpha_k)u(\alpha_k)$$

For $m \to \infty$, the first term on the rhs goes to zero if all eigenvalues of R are smaller than one, and we get

$$v(\alpha_k) = \sum_{i=0}^{\infty} R^i(\alpha_k) u(\alpha_k)$$

Proceeding to policy iteration $k+1$, we have

$$\begin{aligned} v(\alpha_k) &= R(\alpha_k) v(\alpha_k) + u(\alpha_k) \\ &= R(\alpha_{k+1}) v(\alpha_k) + u(\alpha_{k+1}) + \gamma \end{aligned}$$

for some vector

$$\gamma \geq 0$$

since, by definition, α_{k+1} minimizes $R(\alpha)v(\alpha_k)+u(\alpha)$ over α. Iterating again, we obtain

$$\begin{aligned} v(\alpha_k) &= \sum_{i=0}^{\infty} R^i(\alpha_{k+1}) \left(u(\alpha_{k+1}) + \gamma\right) \\ &= v(\alpha_{k+1}) + \sum_{i=0}^{\infty} R^i(\alpha_{k+1}) \gamma \end{aligned}$$

In the standard DP algorithm, the nonnegativity of R ensures that $v(\alpha_k) \geq v(\alpha_{k+1})$. In our case, R as well as $\sum_{i=0}^{\infty} R^i$ contain negative elements[1] and we cannot guarantee that $\sum_{i=0}^{\infty} \left(R^i\right) \gamma$ is positive. However, the row sums of $\sum_{i=0}^{\infty} \left(R^i\right)$ are equal to $1/(1-e^{-\rho \Delta t})$, and the negative elements are small in absolute value compared to the positive row sums. The experience from the test applications suggests that $v(\alpha_k) \geq v(\alpha_{k+1})$ always holds if the method converges at all (except for minimal deviations, which are smaller than the covergence criterion). If this is the case, $v(\alpha_k)$ declines monotonically in k, and convergence is guaranteed if $v(\alpha_k)$ is bounded from below. The latter condition is unproblematic if $u(\alpha)$ is bounded, and the eigenvalues of R are not larger than $1-e^{-\rho \Delta t}$ (those of Π^* not larger than 1). With declining $v(\alpha_k)$, the standard proofs show that the solution to which $v(\alpha_k)$ converges fulfills the HJB equation.

The crucial condition is therefore that the absolute values of the eigenvalues of Π^* are not greater than 1. The next section will discuss how to achieve this. With this condition fulfilled, convergence in policy space is usually achieved in less than 10 iterations.

[1] They are typically in columns that refer to state variables close to the boundary of the state space. In high dimensions, almost all points are close to the boundary of the state space in some dimension, so that almost all columns will contain negative elements, and it is not possible to give a special treatment to points at the boundary.

3.8 Eigenvalues of the transition matrix

The convergence analysis of the previous subsection has supposed that the eigenvalues of $\Pi^*(\alpha)$ are not greater than 1 in absolute values. Since all rows of $\Pi^*(\alpha)$ add up to one (cf. Subsection 3.2), it has a unit eigenvalue, and therefore we can also say that we want the matrix $\Pi^*(\alpha)$ to have a spectral radius of 1. This is true for a stochastic matrix, problems only arise because $\Pi^*(\alpha)$ has some negative entries. However, it turns out that one can give conditions under which the matrix is very likely to satisfy this criterion.

The most important condition is the following. While the state space of the theoretical model may be unbounded, a numerical calculation on a finite grid can obviously represent only a bounded region of the state space. It is desirable to choose this region so that the dynamic system has a tendency to stay within this region. In other words, if we are close to the boundary, the controls should be such as to steer the system towards the centre of the region. It can be shown, at least in a somewhat simplified setting, that this is the crucial requirement to make the transition matrix stable. The Appendix demonstrates this for the case of a one-dimensional grid.

Extensive numerical investigations have shown that this condition is necessary, but not fully sufficient to make the policy iteration scheme stable. The iterations may also become unstable if we try approximations that use too many base functions (for example a cubic rather than a quadratic approximation) in comparison with the number of nearby points chosen for the approximation. The strategy described in Section 3.3 proved to be robust in the test applications.

3.9 Heuristic error bounds

Section 3.7 has established conditions under which we can expect the policy iterations to converge monotonically. In this section, we assume that we have obtained an estimate of the value function and the optimal policy in this way, and we also assume that we had reached the same estimate, by monotonic convergence, if we had started from the true optimal policy (in the test applications this has always been the case). Under these conditions we derive an error bound for the estimated value function. Again, the complexity of the approximation makes it impossible to derive this error bound

strictly; at one point, we will have to invoke a heuristic argument. Nevertheless, this heuristic error bound will provide important insights into the conditions that determine the accuracy of the solution.

Let us first derive an error bound on the estimated cost functional, for a given control α. It is useful to work with the discretized problem for fixed Δt. Asume Δ_t to be very small, so that the discretization error is irrelevant. The analysis only concerns the error resulting from the value function regression, which dominates the discretization error for interesting cases. From now on we distinguish explicitly between the true (except for the discretizaton error) cost functional $v(\alpha)$, which satisfies (6), and the approximate cost functional $w(\alpha)$, satisfying (8). Subtracting (6) from (8), we obtain

$$\begin{aligned} w(\alpha) - v(\alpha) &= e^{-\rho\Delta t}\Pi(\alpha)\left(Q^* w(\alpha) - \begin{bmatrix} v(\alpha) \\ v^{\circ}(\alpha) \end{bmatrix}\right) \\ &= e^{-\rho\Delta t}\left[\Pi(\alpha)Q^*\left(w(\alpha) - v(\alpha)\right) + \Pi(\alpha)\left(Q^* v(\alpha) - \begin{bmatrix} v(\alpha) \\ v^{\circ}(\alpha) \end{bmatrix}\right)\right] \end{aligned} \tag{16}$$

where $\Pi(\alpha) = [\Pi_{11}(\alpha), \Pi_{12}(\alpha)]$ and $Q^* = [I, Q]'$. The task is to derive an upper bound for the supremum norm of this difference. In a first step, however, we make use of a different vector norm, defined as

$$||x||_{\alpha} = \sqrt{\left(\Omega^{-1}x\right)^H \left(\Omega^{-1}x\right)} \tag{17}$$

Here, H denotes Hermitian transpose, and Ω is the matrix whose columns are the eigenvectors of $\Pi(\alpha)Q^*$, all normalized to unit length. This norm is similar to the Euclidean norm, computed with the coefficients of vector x w.r.t. the basis formed by the normalized eigenvectors of $\Pi(\alpha)Q^*$. (From a different point of view, (17) is the Euclidian norm in a linear space where coefficients are with respect to the basis Ω^{-1}.)

We can now apply this norm to (16). Obviously, if all the eigenvalues of the matrix $\Pi(\alpha)Q^*$ are smaller than 1 in absolute value, $||\Pi(\alpha)Q^*x||_{\alpha} \leq ||x||_{\alpha}$ and therefore, using the triangle inequality,

$$||w(\alpha) - v(\alpha)||_{\alpha} \leq e^{-\rho\Delta t}\left[||w(\alpha) - v(\alpha)||_{\alpha} + \left|\left|\Pi(\alpha)Q^* v(\alpha) - \Pi(\alpha)\begin{bmatrix} v(\alpha) \\ v^{\circ}(\alpha) \end{bmatrix}\right|\right|_{\alpha}\right]$$

and therefore

$$||w(\alpha) - v(\alpha)||_\alpha \leq \frac{e^{-\rho\Delta t}}{1 - e^{-\rho\Delta t}} \left|\left| \Pi(\alpha)Q^* v(\alpha) - \Pi(\alpha) \begin{bmatrix} v(\alpha) \\ v^\circ(\alpha) \end{bmatrix} \right|\right|_\alpha \tag{18}$$

The dependence of the norm on α is awkward; it would be very useful if we could translate this inequality into an equality in the supremum norm. However, this is not strictly possible: while it can be easily shown that $||x||_{sup} \leq \sqrt{n}\,||x||_\alpha$, the supremum norm does not provide an upper bound on the α norms that holds *for all* α. Here we have to rely on a heuristic argument which is based on computational experience and which takes into account the characteristics of the vectors that appear on both sides of (18): the vector on the rhs of (18) is always dominated by a constant, since the absolute level of the value function can be much less precisely estimated than differences. Then, since the constant vector is an eigenvector of $\Pi(\alpha)Q^*$, the corresponding coefficient is several orders of magnitude greater than all other coefficients, and in that case, it is easily shown that $||x||_{sup} \approx ||x||_\alpha / \sqrt{N}$. The vector on the rhs of (18) has no special relationship to the eigenvectors of $\Pi(\alpha)Q^*$, so that the coefficients are more or less "randomly" distributed, having the same order of magnitude, and in that case $||x||_{sup}$ always tends to be greater than $||x||_\alpha / \sqrt{N}$. If this is true, it is safe to rewrite (18) with supremum norms, which gives

$$\begin{aligned} ||w(\alpha) - v(\alpha)||_{sup} &\leq \frac{e^{-\rho\Delta t}}{1 - e^{-\rho\Delta t}} \left|\left| \Pi(\alpha)Q^* v(\alpha) - \Pi(\alpha) \begin{bmatrix} v(\alpha) \\ v^\circ(\alpha) \end{bmatrix} \right|\right|_{sup} \\ &\leq \frac{e^{-\rho\Delta t}}{1 - e^{-\rho\Delta t}} \left|\left| Q^* v(\alpha) - \begin{bmatrix} v(\alpha) \\ v^\circ(\alpha) \end{bmatrix} \right|\right|_{sup} \\ &= \frac{e^{-\rho\Delta t}}{1 - e^{-\rho\Delta t}} \epsilon(\alpha) \end{aligned} \tag{19}$$

with $\epsilon(\alpha)$ defined accordingly.

To derive an upper bound on the deviation of the estimated optimal cost functional (value function) from the exact value function, let us define α^* as the true optimal control, and $\hat{\alpha}$ as the estimated optimal control, to which the algorithm has converged.

Then we have

$$v(\alpha^*) \leq v(\hat{\alpha}) \tag{20}$$

$$w(\hat{\alpha}) \leq w(\alpha^*) \tag{21}$$

where the first equality is by definition of v, and the second follows from the assumption of monotonic convergence from the true solution. From equations (20) and (21) and (19) for α^* and $\hat{\alpha}$, we obtain for each component n of the cost vectors

$$v_n(\alpha^*) \leq w_n(\hat{\alpha}) + \frac{e^{-\rho\Delta t}}{1 - e^{-\rho\Delta t}}\epsilon(\hat{\alpha})$$
$$w_n(\hat{\alpha}) \leq v_n(\alpha^*) + \frac{e^{-\rho\Delta t}}{1 - e^{-\rho\Delta t}}\epsilon(\alpha^*)$$

Combining them gives an upper bound on the deviation

$$||v(\alpha^*) - w(\hat{\alpha})||_{sup} \leq \frac{e^{-\rho\Delta t}}{1 - e^{-\rho\Delta t}} \max\{\epsilon(\alpha^*), \epsilon(\hat{\alpha})\} \tag{22}$$

It remains to find a more useful expresson for the ϵ's. Equation (13) provides a relationship between the step size h_i and time Δt. Somewhat simplified, we have

$$h_i = \begin{cases} |b_i| \cdot d \cdot \Delta t & \text{if } b \text{ dominates} \\ \sqrt{\sigma_i^2 d \cdot \Delta t} & \text{if } \sigma^2 \text{ dominates} \end{cases} \tag{23}$$

If $\bar{b}$ and $\bar{\sigma}$ are some upper bounds on derivatives and variance (possible depending on x), we can use (9) to obtain the bounds

$$\epsilon(\alpha) \leq d \cdot \Delta t \cdot \max\left\{ \left|\left| v_x^{err}(\alpha) \cdot \bar{b} \right|\right|_{sup}, \tfrac{1}{2} \left|\left| v_{xx}^{err}(\alpha) \cdot \bar{\sigma}^2 \right|\right|_{sup} \right\} \tag{24}$$

where $v_x^{err}(\alpha)$ and $v_{xx}^{err}(\alpha)$ are the approximation errors of the gradient and Hessian of $v(\alpha)$ for given regression functions. Inserting into (22), we obtain for small Δt

$$||v(\alpha^*) - w(\hat{\alpha})||_{sup} \leq \frac{d}{\rho} \max\Big\{ \left|\left| v_x^{err}(\alpha^*) \cdot \bar{b} \right|\right|_{sup}, \tfrac{1}{2} \left|\left| v_{xx}^{err}(\alpha^*) \cdot \bar{\sigma}^2 \right|\right|_{sup}, \left|\left| v_x^{err}(\hat{\alpha}) \cdot \bar{b} \right|\right|_{sup}, \tfrac{1}{2} \left|\left| v_{xx}^{err}(\hat{\alpha}) \cdot \bar{\sigma}^2 \right|\right|_{sup} \Big\} \tag{25}$$

The error bound in equation (25) depends on the estimation error of derivatives for the true value function, and of the true cost function for the estimated control. Even if the true value function is well approximated by the regressors so that $\epsilon(\alpha^*)$ is small, the

value function for some bizarre choice of α may not be well approximated, and $\epsilon(\alpha)$ may be large. This again underlines the necessity to start from a reasonable specification of α, so that the corresponding cost functional in each policy iteration step is well approximated.

The numerical tests in Section 4 will demonstrate the usefulness of equation (25).

3.10 Computational issues

The code was written in ANSI C, and executed on a Pentium 200 PC, using the Linux operating system and the GNU C-compiler.

The main computational problem is the solution of the large linear system (8) in each policy iteration step. The matrix $\Pi^*(\alpha)$ is sparse, but not very sparse: Each row contains as many non-zero elements as nearby points are used for regression. In the largest application so far, I had 8000 rows with 200 non-zero elements in each row. State of the art are conjugate-gradient methods, for which C-code is available from Barrett, Berry, Chan, Demmel, Donato, Dongarra, Eijkhout, Pozo, Romine and van der Vorst (1993). I used the GMRES method (Barrett et al. 1993, p.19), which seems preferable for not very sparse systems of moderate dimension, since the storage requirement is proportional to the dimension, so that the method need not be restarted very often. Using a good preconditioner is essential; the D-ILU preconditioner (Barrett et al. 1993, p.45) appears to work very well. The above system of 8000 linear equations can be solved on the PC in about 80 seconds. To speed up the process one could apply simple Coarse Grid – Fine Grid schemes (I could not yet implement a full multigrid method), but since the GMRES method works so well, it does not seem worth the effort.

4 Test application

4.1 Design of the test

The heuristic error analysis has shown that the accuracy of the estimated value function will mainly depend on how close the true solution is to the span of the approximating functions. The quality of the approximating functions in turn depends on our prior information about the model, and the question "What is the precision this method can

achieve for a given model?" is therefore not a very meaningful one. This open character of the method makes it difficult to evaluate its accuracy. For a test model with known analytic solution, we must "ignore" this information, in order to get a reasonable test of the accuracy of the method.

It is therefore more important to look at the qualitative properties of the method for a range of models with differing degrees of nonlinearity. The tests should provide evidence on whether the heuristic analysis of convergence and error bounds is actually valid. In particular, the test applications are designed to answer the following questions:

1. Under what conditions does the method converge? How stable is the convergence?
2. Is the end result sensitive to starting values of the policy function?
3. Are the heuristically derived error bounds valid?
4. If the value function is not well approximated by the base functions, does the method provide ways to find better base functions?
5. How does noise affect the accuracy of the solution?
6. How does nonlinearity affect the accuracy of the solution?
7. How does the accuracy vary with the number of dimensions and number of grid points?

4.2 Test model

The starting point is the simplest model available, the linear quadratic control problem, where the quadratic objective

$$\mathrm{E}\left[\int_0^\infty e^{-\rho t}\left(x'Qx + u'Ru\right)\,dt\right]$$

is minimized over the control vector u s.t. the linear dynamic equation

$$dx = (Ax + Bu)dt + \sigma\, dz$$

It is well known that the optimal control u is linear in x, and the value function is quadratic in x:

$$v(x) = x'Vx + v_0, \qquad u(x) = Ux \tag{26}$$

for some matrices V and U and scalar v_0. To serve as a test problem, this simple problem has to be transformed into one that is difficult to solve for the numerical method. This can be done by operating on a different state space. Define

$$y = T(x) \tag{27}$$

where $T : \mathcal{R}^d \to \mathcal{R}^d$ is any invertible, twice continuously differentiable vector function. In the new state space, the original problem (1) and (2) is transformed into the equivalent problem to minimize

$$\mathrm{E}\left[\int_0^\infty e^{-\rho t}\left[\left(T^{-1}(y)\right)' Q T^{-1}(y) + u' R u\right]\, dt\right]$$

s.t. (cf. Ito's lemma, Kushner and Dupuis, 1992, p.13)

$$dy_i = \left[DT_i \cdot \left(AT^{-1}(y) + Bu\right) + \frac{1}{2}\,\mathrm{tr}\left(D^2 T_i \cdot \sigma\sigma'\right)\right] dt + D\,T_i \cdot \sigma\, dz \tag{28}$$

where $D\,T_i$ and $D^2\,T_i$ are the Jacobian and Hessian of the i-th component of T, respectively. In terms of y, the solution is given by

$$v(y) = T^{-1}(y)' V T^{-1}(y) + v_0, \qquad u(y) = U T^{-1}(y) \tag{29}$$

While this problem is very simple in x-space, the dynamic equation, the cost function, the value function and the optimal control in y-space may be highly nonlinear, depending on the chosen specification of T. Critical for the present method is the functional form of the value function, compared to the regressors used. One aspect of the original solution remains: the control is independent of the level of noise. But I think that this does not facilitate the numerical solution in any way.

The test applications use the following numerical specification: The discount factor is $\rho = 0.05$, so that the time period conforms roughly to one year in economic models. To be able to calculate the exact solution as accurately as possible, I start from a diagonal system, where B, Q and R are identity matrices, and the diagonal entries of A are between -0.2 and -1, so that the components of the system tend to return to the steady steady within a time span of 1 to 5 periods. The diagonal system is subject to a composition of three transformations:

$$T(x) = F(\Lambda(G(x))) \tag{30}$$

where F and G are constant unitary matrices (eigenvalues on the unit circle), and Λ is a diagonal nonlinear transformation. That means, the diagonal system is first rotated, then subjected to a component-wise nonlinear transformation, and (in some cases) rotated again. The benchmark case of the next section uses

$$\Lambda_i(x) = (x + k_i)^{p_i}, \qquad i = 1, \ldots, d \tag{31}$$

where k_i is some constant, and $F = I_d$ (the system is rotated only once). The powers p_i are equally distributed between 0.5 and 2; with two dimensions, $p_1 = 2$ and $p_2 = 0.5$, with four dimensions, $p_1 = 2$, $p_2 = 1.5$,$p_1 = 1$, $p_2 = 0.5$ etc.

In all applications, the state space was chosen as a hypercube in y-space such that it approximately covers the hypercube $[-1, 1]^d$ in x-space. In a linear-quadratic model, the choice of scale is relevant only for considering what is a "large" level of noise. Large must be understood as in comparison to the region on which the model is solved. The applications use values of σ up to 0.3, which means that, in each dimension, the region that the system can reach within one period with probability 0.95 covers 60 percent of the state space.

4.3 Results

In order to evaluate the accuracy achieved with the VFR method, it is useful to compare it to the results obtained from a more conventional method. The policy iteration scheme of Kushner and Dupuis (1992) with locally consistent discrete approximation as described in Section 2.2 can probably be considerd as the standard method to handle continuous time stochastic control problems. In the usual form it cannot be applied to higher-dimensional problems. I therefore apply it to a 2-dimensional problem, and then ask whether VFR can achieve the same accuracy for problems of up to 10 dimensions. The problems were standardized so that a meaningful comparison of the accuracy between applications in different dimensions is possible.

Table 1 provides the results of the standard method for the 2-dimensional benchmark case with different grid sizes. The first line corresponds to the solution on a grid of 10000 (100×100) points with a standard deviation of noise in x-space of $\sigma = 0.1$ for both variables. The following lines describe the results for different pa-

rameter combinations, whereby parameters that are not listed conform to the specification in the first line. To measure accuracy, the maximal deviation of the solution from the exact solution is provided, more precisely $\left|\left|v^{exact} - v^{appr}\right|\right|_{sup} / \left|\left|v^{exact}\right|\right|_{sup}$ and $\max_{i=1}^{d} \left\{ \left|\left|u_i^{exact} - u_i^{appr}\right|\right|_{sup} / \left|\left|u_i^{exact}\right|\right|_{sup} \right\}$. The supremum was taken over all grid points, including those at the boundary of the state space, since the accuracy was equally good at the boundary. The results show that the accuracy increases systematically in N, but slowly: an increase in N by a factor of 100 is necessary to increase accuracy by a factor of about 10. No systematic relationship between σ and accuracy appears.

Parameters		Maximal deviation	
N	σ	v	u
10000	0.1	2.12e-02	6.38e-03
2500		4.33e-02	1.30e-02
900		7.42e-02	2.21e-02
100		3.01e-01	6.68e-02
	0.0	1.05e-02	9.44e-03
	0.2	2.33e-02	3.40e-03

Table 1: Results for standard method, 2-dimensional benchmark model

Table 2 provides results for VFR on a Sobol grid for several variants of the 6-dimensional model.[2] The parameters are N, the number of Sobol points, the standard deviation σ (the same for all variables) in x-space, and "Rep.", giving the number of repetitions of the algorithm to improve the regressors, cf. Step 8 in Section 3.6 and Section 3.3. The entry "0+" means that third order polynomial terms were included, but no search over nonlinear transformation performed. As functions $f(.)$ I simply used non-negative powers of the individual series. Working in y-space, this means

$$f_i(y_i) = y_i^{r_i} \tag{32}$$

If the transformations of the state space are of the form (31) and $F = I$, the correct choice of r_i (namely $r_i = 1/p_i$) will obviously make the true value function quadratic

[2] I have also run part of the models using generalized Faure points, but I could not detect a systematic advantage of either type of sequences, and I do therefore not report the results.

Parameters			Maximal deviation				
N	σ	Rep.	v	D v	D^2 v	u	EBound
1) Quadratic value function							
200	0.1	0	4.97e-03	3.50e-04	3.61e-04	6.65e-04	1.54e-12
2) Powers 0.2–5, exact base functions							
200	0.1	0	4.54e-03	2.50e-04	2.51e-04	6.43e-04	2.84e-08
3) Benchmark							
1000	0.1	2	3.76e-02	6.97e-03	1.13e-02	9.99e-03	2.20e-01
		0	6.82e-01	8.94e-02	1.78e-01	8.94e-02	3.38e-00
		0+	1.49e-01	1.75e-02	3.09e-02	2.51e-02	4.32e-01
		1	7.07e-02	1.07e-02	1.79e-02	1.53e-02	3.34e-01
		5	7.89e-04	1.72e-03	2.72e-03	2.46e-03	7.39e-02
		10	4.96e-03	4.36e-04	7.97e-04	1.04e-03	9.06e-03
200			7.08e-02	1.17e-02	2.52e-02	2.81e-02	7.69e-01
500			1.41e-02	8.22e-03	1.36e-02	1.61e-02	4.85e-01
1500			1.44e-02	3.77e-03	6.34e-03	5.41e-03	2.41e-01
	0.0		6.07e-02	6.99e-03	1.11e-02	1.00e-02	3.02e-01
	0.2		1.56e-02	6.76e-03	1.13e-02	9.70e-03	1.22e-01
	0.3		3.54e-03	6.32e-03	1.05e-02	9.29e-03	7.07e-02
4) Powers 0.25–4							
1000	0.1	2	5.28e-01	3.92e-02	6.43e-02	6.22e-02	2.82e-01
		0	1.39e-00	1.71e-01	3.66e-01	1.88e-01	4.31e-00
		10	3.67e-02	2.98e-03	5.37e-03	4.72e-03	1.74e-02
5) Logistic 0.05-0.2							
1000	0.1	2	1.63e-01	1.58e-02	2.83e-02	1.86e-02	1.47e-00
		0+	1.67e-01	1.55e-02	2.86e-02	1.86e-02	1.47e-00
6) Logistic 0.1–0.4							
1000	0.1	2	5.79e-01	5.66e-02	1.11e-01	6.07e-02	5.54e-00
		0+	5.92e-01	5.71e-02	1.13e-01	6.06e-02	5.40e-00
7) Powers 0.7–1.5, F unitary							
1000	0.1	2	1.48e-00	2.85e-01	4.11e-01	3.04e-01	6.75e-00
		0+	4.56e-01	1.33e-01	2.07e-01	1.42e-01	4.27e-00
8) Logistic 0.1–0.4, F unitary							
1000	0.1	2	1.41e-00	1.53e-01	5.42e-01	2.98e-01	1.75e+01
		0+	1.10e-00	1.51e-01	3.74e-01	2.73e-01	1.44e+01

Table 2: Results for VFR method, 6-dimensional model

in $f(y)$. The question is whether the method can endogenously find out approximately correct values for the r_i.

The accuracy measures provided are the same as in Table 1, and in addition supremum norms are given for the error in the estimated first and second derivative of the value function. The derivative was always computed in x-space, not in y-space, since the Hessian in x-space is constant, which allows a better interpretation of the results and comparison across different transformations T. More precisely, the errors are

$$\max_{i=1}^{d} \left|\left| \frac{\partial v^{exact}}{\partial x_i} - \frac{\partial v^{appr}}{\partial x_i} \right|\right|_{sup} / \max_{i=1}^{d} \left|\left| \frac{\partial v^{exact}}{\partial x_i} \right|\right|_{sup}$$

$$\max_{i=1}^{d} \max_{j=1}^{d} \left|\left| \frac{\partial^2 v^{exact}}{\partial x_i \partial x_j} - \frac{\partial^2 v^{appr}}{\partial x_i \partial x_j} \right|\right|_{sup} / \max_{i=1}^{d} \max_{j=1}^{d} \left\{ \frac{\partial^2 v^{exact}}{\partial x_i \partial x_j} \right\}$$

The supremum was taken over a (pseudo-) random sample of $\max(1000, N)$ points from the inner part of the state space (corresponding to the hypercube $[-0.7, 0.7]^d$ in x-space). These points are *not* part of the low discrepancy grid on which the solution was developed, but obtained from the value function on the original grid by the same approximation scheme that was used in the policy iteration. The last column of Table 2 provides a tightened version of the error bound (25), based on the estimated errors of α^* and using the exact values of b and σ rather than any bounds. Again, derivatives were taken in x-space, not y-space.

Cases 1) and 2) illustrate the maximum accuracy the method can achieve, namely when the value function is actually spanned by the regressors. Case 1) is the simplest one, where Λ is the identical transformation, so that the value function is quadratic in y. The approximation error for the derivatives of v is practically zero, which is reflected in an error bound of about 1e-12. Due to a finite Δt the actual deviation is of course greater, but it is about one order of magnitude smaller than for the standard method in 2 dimensions. This is achieved with only 200 data points in 6 dimensions. Case 2) introduces strong nonlinearity in the transition equation, by using exponential transformations varying from 0.2 to 5 for the different variables. However, using as regressors quadratic functions in the appropriate powers[3] of y, the regressors span the value function and the accuracy is the same as in Case 1). This justifies the claim

[3]In addition, I transformed the Sobol grid so that points are evenly distributed in y^p-space, not in y-space. This is another way to improve the accuracy of the method.

that the accuracy is not affected by nonlinearity in the dynamic equation etc., what counts is only how well the value function is approximated by the regressors. For all the calculations I have used $\Delta_t = 0.00001$. It is in fact possible to increase the accuracy in cases 1) and 2) by a factor of more than ten by using even smaller Δt, but it turns out that this affects accuracy negatively in other cases, probably due to roundoff error problems. So I stick to the above specification.

Case 3) is the benchmark case, with exponential transformations of the state variables, where initially the only regressors are second order polynomials in y. If no repetitions are performed to improve the regressors, the approximation is accordingly poor, as can be seen from the second line of Case 3). Increasing the number of iterations, the accuracy increases constantly, and for 10 iterations it is almost as good as in Cases 1) and 2). Fixing the number of repetitions at 2, the next lines show that the accuracy increases steadily with the number of grid points. Finally, varying the degree of stochastics, we see that the accuracy *increases* slightly with the level of noise. This can probably be explained by the fact that noise has a tendency to smooth the solution, so that it can be better approximated by smooth regressor functions. Case 4) is very similar, but with even higher degree of nonlinearity. Finally the same accuracy is reached, but only after more repetitions. These examples show that VFR, for a problem of 6 dimensions and a grid of 1000 points, can reach a level of accuracy as high as the standard method for 2 dimensions on a grid of 10000 points. To achieve this, we need to have a good idea about the functional form of the value function.

Cases 5)–8) differ from the foregoing cases in that the value function is not in the span of the regressors, not even if we use the optimal values of the r_i's. Case 5) applies the logistic transformation

$$\Lambda_i(x) = \frac{a_i}{1 + b_i e^{-c_i}} \qquad i = 1, \ldots, d \tag{33}$$

with values of c_i between 0.05 and 0.2. Since the logistic function has a turning point, functions of the form (32) are of no help, and cases 5) and 6) show that the attempt to improve the approximation is useless. Correspondingly, the accuracy decreases considerably.

Cases 7) and 8) use $F \neq I$, so that y is a linear combination of nonlinear transformations, which cannot be reversed by looking for suitable transformations on individual

variables. The accuracy of the method decreases even more, and interestingly, the attempt to improve the regressors can make things worse, as demonstrated by case 7). The error bound in both cases shows that the value function is very poorly approximated by the regressors. I have made no further effort to improve the regressors; important about cases 7) and 8) is not so much the accuracy achieved, but the fact that even when the approximation is so poor that it is almost useless, the method still converges. This indicates that accuracy, and not the stability of the policy iteration, is what actually limits the applicability of the method. I have to add, however, that the method may well break down if the regressors provide a too bad approximation to the value function. This happens, for example, with transformations (31) and powers from 0.2 to 5 and a second order polynomial approximation in y.

Interesting from Table 2 is also the close correlation between the calculated error bound and the actual sup-norm deviation of v. The latter one is always smaller by a factor between 2 and 10, except in the cases where the value function is exactly spanned. This confirms that (25) is the key equation to understand the accuracy of the method.

Summarizing the above results, we can answer the questions posed in Section 4.1. First and most importantly, the method shows stable convergence behavior, even in cases where the true value function is poorly approximated by the regressors. The starting point of the iterations is not too crucial: all the above results were obtained by starting from a linear approximation of the controls in y-space, which conforms to what we get from a linear-quadratic approximation about the stationary state. In should always be possible to get starting values at least as good as that. Starting the results from the exact solution leads to virtually identical results (which are therefore not reported). Moreover, the estimated value function that we obtain contains information that can be used to find better regressors, and to iteratively improve the accuracy of the solution. This may often require a more sophisticated analysis of the obtained solution than the one employed here, but the important point is that the estimated value function contains this information.

The results also show that a higher level of noise does not negatively affect the accuracy of the solution, the comparative advantage of the method is therefore in the solution of models with significant amount of noise. Nonlinearity of the dynamic equation is no

problem. Important is only whether good approximating functions are available for the true value function.

Finally I present results that should shed some light on question 7. As I said in the Introduction, I cannot make any theoretical claim relating to the curse of dimensionality. It might nevertheless be useful to have at least some preliminary evidence on the additional computational effort that is necessary to deal with more state variables. Figure 1 displays the accuracy for the benchmark model (maximal deviation of u) in 2, 4, 6, 8 and 10 dimensions for grid sizes of 63, 125, 250, 500, 1000, 2000, 4000 and 8000. Roughly speeking, it seems that an increase of 2 in the number of dimensions requires about a fourfold increase in the number of grid points, but one should be aware that these numbers are probably strongly model-dependent. The change in the computational burden is of course higher, since the number of nearby points is increasing quadratically in the dimension. To cite just one example, the change from 6 dimensions and 1000 grid points to 8 dimensions and 4000 grid points increases computation time by a factor of about 12.

5 Conclusions

This paper has explored the use of value function regression for solving continuous time stochastic dynamic programming problems, and has given some test applications. The applications show that smooth problems with up to about 10 state variables can be accurately solved on a PC. This conclusion is also supported by former tests with disaggregated stochastic growth models (Reiter 1997).

While the overall method is rather complicated, most of the work can be done by library functions. The specific application has to compute the derivatives of the state variables, for given controls, and the optimal control variables, given the value function and its derivatives. More work has to be done only if regressor functions must be provided that are particularly suitable for the specific application. In my experience, this is the only step where "babying" of the method might be necessary. If suitable regressors are found, the method convergences without problems and in predictible time. No nonlinear optimization or equation solving is needed.

A further advantage of the method is that restrictions on state and control variables

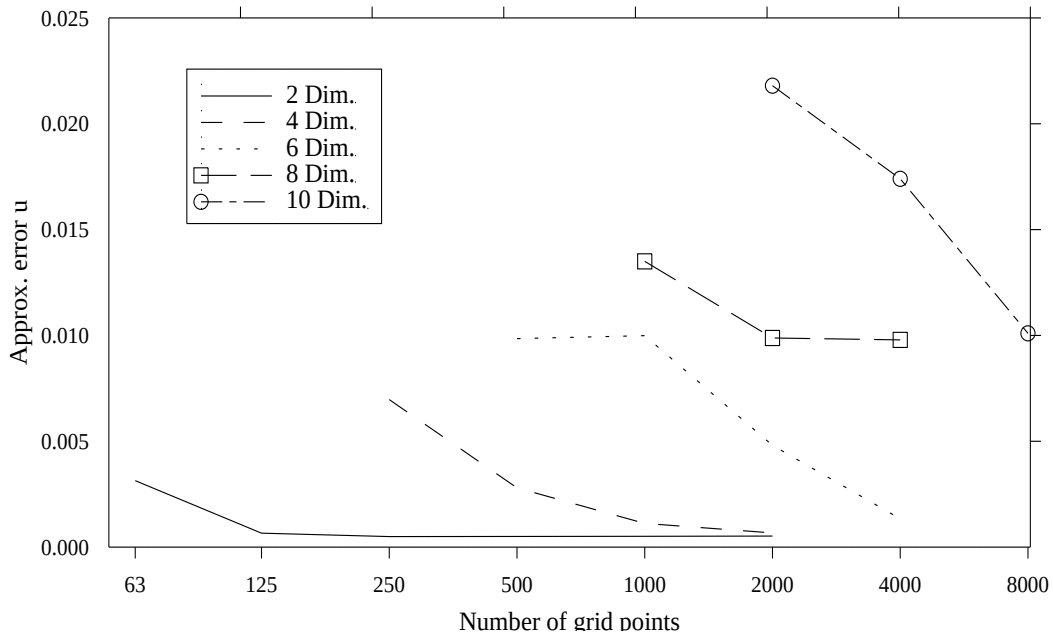

Figure 1: Accuracy in benchmark model, 2 to 10 dimensions

can be handled in the same straightforward way as in the standard Bellman algorithm, with the following qualification: restrictions on control variables usually lead to a value function with discontinuous first derivatives, which cannot be very well approximated by smooth functions. Experimenting with non-smooth approximating functions is left for future work.

A Eigenvalue analysis in a simple one-dimensional case

A.1 Preliminaries

In the following, we will encounter matrices of the form

$$f_1 a_1' + f_2 a_2' \tag{34}$$

where the f_i's and a_i's are arbitrary column vectors (of same size) and the prime denotes transposition. Obviously, this matrix has rank at most two, and therefore not more than two nonzero eigenvalues. To obtain the nonzero eigenvalues, we try eigenvectors of the form $(\mu_1 f_1 + \mu_2 f_2)$ and some algebra shows that the eigenvalues are

$$\lambda_{1,2} = \frac{1}{2}\left(a_1' f_1 + a_2' f_2 \pm \sqrt{(a_1' f_1 - a_2' f_2)^2 + 4 a_1' f_2 \cdot a_2' f_1}\right)$$

which reduces to $a_1' f_1$ and $a_2' f_2$ if one of the cross terms $a_1' f_2$ or $a_2' f_1$ is zero.

A.2 Eigenvalue analysis

The eigenvalue analysis in the general case is obviously very difficult, if not impossible. Therefore, I restrict myself to the one-dimensional problem on a uniform grid, and investigate the case where the quadratic approximation is based on all grid points, not only on the nearest points. The qualitative conclusions from this analysis seem to be borne out by the more complex practical applications in this paper.

The grid has size N, the grid points are $x^n, \quad n = 1, \ldots, N$. W.l.o.g. we can normalize the state space so that $\sum_n x^n = 0$ and the difference between grid points is equal to 1. We use the quadratic approximation:

$$\hat{V}(x) = \beta_0 + \beta_1 x + \beta_2 x^2 \tag{35}$$

The parameter estimates $\hat{\beta}$ are then given by

$$\begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \hat{\beta}_2 \end{pmatrix} = Bv, \qquad B = \left(Z'Z\right)^{-1} Z', \qquad Z = (e, X, X.X)$$

where X is the column vector of grid points, e is a column vector of ones, v is the value function vector, and the lower dot means element-by-element multiplication of column vectors. B is a $3 \times N$ matrix, for which we obviously have

$$BZ = I \tag{36}$$

The matrix $\Pi^* = \Pi_{11} + \Pi_{12}Q$ can be written

$$\begin{pmatrix} P^1 & & 0 \\ & \ddots & \\ 0 & & P^n \end{pmatrix} \begin{pmatrix} Q^1 \\ \ddots \\ Q^n \end{pmatrix} \tag{37}$$

where $P^n = (p_{n0}, p_{n-}, p_{n+})$ is a 1 by 3 matrix containing the probabilities that the system stays at x^n or switches to $x^n - h$ or $x^n + h$. The $3 \times N$-matrices Q^n are obtained from the approximation

$$V(x^n \pm h) = V(x^n) \pm V'(x^n)h + 0.5V''(x^n)h^2 \tag{38}$$

where $V(x^n)$ is known since x^n is a grid point, and the first and second derivatives V' and V'' are based on (35). More explicitly, the block Q^n is

$$\begin{bmatrix} e_n' \\ e_n' - \left(B_1' + 2B_2'x^n\right)h + B_2'h^2 \\ e_n' + \left(B_1' + 2B_2'x^n\right)h + B_2'h^2 \end{bmatrix} \tag{39}$$

where B_0', B_1' and B_2' denote the rows of B, and e_n' is a row vector of zeros, except of element n which is equal to one. Inserting (39) into (37) we obtain

$$\Pi^* = I + h\left(P_+ - P_-\right)B_1' + \left\{2h\left[\left(P_+ - P_-\right).X\right] + h^2\left(P_- + P_+\right)\right\}B_2' \tag{40}$$

where P_- and P_+ are column vectors with elements p_{n-} and p_{n+}, respectively. In light of (3) and (4) we have $(P_+ - P_-) = h \cdot f(x, \alpha)\phi$ and $(P_- + P_+) = \sigma^2\phi$ where the factor

$\phi = 1/(\max_x \{\sigma^2 + hf(x,\alpha)\})$ is almost independent of h for very small values of h. Then we can write (40) as

$$\Pi^* = I + \left[h^2\phi f(x,\alpha)\right] B_1' + h^2\phi \left[2f(x,\alpha).X + \sigma^2\right] B_2'$$

The matrix Π^* is of the form (34). Defining $f_1 = h^2\phi f(x,\alpha)$, $f_2 = h^2\phi \left[2f(x,\alpha).X + \sigma^2\right]$ $a_1' = B_1'$ and $a_2' = B_2'$, we see that the eigenvalues of Π^* are given by

$$\lambda_{1,2} = 1 + \frac{1}{2}\left(a_1'f_1 + a_2'f_2 \pm \sqrt{(a_1'f_1 - a_2'f_2)^2 + 4a_1'f_2 \cdot a_2'f_1}\right)$$

A set of sufficient conditions for $|\lambda_{1,2}|$ to be smaller than 1 is then

$$-1 < a_1'f_1 < 0 \tag{41a}$$

$$-1 < a_2'f_2 < 0 \tag{41b}$$

$$a_1'f_1 \cdot a_2'f_2 > a_1'f_2 \cdot a_2'f_1 \tag{41c}$$

$$a_1'f_1 \cdot a_2'f_2 > -1 \tag{41d}$$

Since f_1 und f_2 decrease quadratically in h, conditions (41d) as well as the first inequalities in (41a) and (41b) are easily met by choosing h small enough (this is not restrictive). Crucial are the sign restrictions in (41a) and (41b), and condition (41c). The nature of these conditions can be most easily seen by considering cases where the time derivative $f(x,\alpha)$ depends linearly on x, in particular the case $f(x,\alpha) = ax$. Since the center of our state space was normalized to zero, this means that x tends to the central value if $a < 0$, and tries to leave the state space if $a > 0$. Using (36) we get

$$a_1'f_1 = h^2\phi B_1' f(x,\alpha) = h^2\phi a B_1' X = h^2\phi a \tag{42}$$

$$a_1'f_2 = h^2\phi B_1' \left[2f(x,\alpha).X + \sigma^2\right] = h^2\phi \left[2aB_1'(X.X) + \sigma^2 B_1' e\right] = 0 \tag{43}$$

$$a_2'f_1 = h^2\phi B_2' f(x,\alpha) = h^2\phi a B_2' X = 0 \tag{44}$$

$$a_2'f_2 = h^2\phi B_2' \left[2f(x,\alpha).X + \sigma^2\right] = h^2\phi \left[2aB_2'(X.X) + \sigma^2 B_2' e\right] = 2h^2\phi a \tag{45}$$

and see that conditions (41) are met if and only if $a < 0$, which conforms to the result stated in Section 3.8: The control must be chosen such that the state variable tends to return to the mid-point $x = 0$.

The analysis reveals another interesting aspect of the approximation scheme: The matrix $\Pi^* - I$ is of rank two, it maps into zero all those vectors that make a zero

contribution to β_1 and β_2. These vectors form a $(n-2)$-dimensional subspace. The matrix $\left(I - e^{-\rho\Delta t}\Pi^*\right)^{-1}$, which transforms the revenue into the value function, therefore magnifies by $1/(1-e^{-\rho\Delta t})$ all those components that make a zero contribution to β_1 and β_2. The intuitive explanation is that those components have an estimated first and second derivative of zero, the component is estimated as constant and therefore multiplied by $1/(1-e^{-\rho\Delta t})$.

References

Barrett, R., Berry, M., Chan, T., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C. and van der Vorst, H.: 1993, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, Siam Publications, Philadelphia, PA.

Judd, K. L.: 1992, Projection methods for solving aggregate growth models, *Journal of Economic Theory* **58**(2), 410–52.

Judd, K. L.: 1996, Approximation, perturbation, and projection methods in economic analysis, *in* H. M. Amman, D. A. Kendrick and J. Rust (eds), *Handbook of Computational Economics*, Elsevier, Amsterdam.

Keane, M. P. and Wolpin, K. I.: 1994, The solution and estimation of discrete choice dynamic programming models by simulation and interpolation: Monte Carlo evidence, Federal Reserve Bank of Minneapolis, Staff Report 181.

Kushner, H. J. and Dupuis, P. G.: 1992, *Numerical methods for stochastic control problems in continuous time*, Springer, London and Tokyo.

Papageorgiou, A. and Traub, J.: 1996, Beating Monte Carlo, *Risk* **9**(6), 63–5.

Paskov, S. H.: 1994, Computing high dimensional integrals with applications to finance, Columbia University, Dept. of Computer Science, Techn. Rep. CUCS-023-94.

Press, W., Flannery, B., Teukolsky, S. and Vetterling, W.: 1986, *Numerical Recipes*, Cambridge University Press.

Reiter, M.: 1997, Solving higher-dimensional continuous time stochastic control problems by value function interpolation, University of Munich, Working Paper version.

Rust, J.: 1996, Numerical dynamic programming in economics, *in* H. M. Amman, D. A. Kendrick and J. Rust (eds), *Handbook of Computational Economics*, Elsevier, Amsterdam.

Rust, J.: 1997a, A comparison of policy iteration methods for solving continuous-state, infinite-horizon Markovian decision problems using random, quasi-random, and deterministic discretizations, Yale University.

Rust, J.: 1997b, Using randomization to break the curse of dimensionality, *Econometrica* **65**(3), 487–516.