Chapter 3

Essay II

Abstract

This work focuses on lightening the curse of dimensionality in a range of to multi-dimensional dynamic programming problems. Discretization in state space and iterative methods are used to find the optimal value function. The differentiability of the optimal value function is utilized to solve the two-stage discrete Hamilton-Jacobi-Bellman equation.

The procedure suits for a wide range of optimal control problems in resource economics. Its efficiency is exemplified by the solving of a few simple problems.

3.1 Introduction

Dynamic programming (DP) is a popular technique for the solving of optimal control problems and stems from the early contribution of Bellman (1957). The technique, which is built upon the Hamilton-Jacobi-Bellman equation, provides a mathematical formalization of the trade-off between current and future profit. Unlike trajectorywise approaches such as direct discretizations or methods based on the Pontryargin's maximum principle, it gives a global solution to the problem stated. It is well known, however, that for many problems the computational requirements are so overwhelming that DP is considered unsuitable.

Nevertheless, the concept of dynamic programming is more powerful than many scientists seem to realize. A wide range of problems in higher state and control space dimensions are solvable with DP, and extensive work in the field of operational research has been undertaken to overcome the curse of dimensionality (see Rust (1996) for an overview of computational complexity). Methods like neuro-dynamic programming (Bertsekas and Tsitsiklis 1995), higher-order approximations, randomization and adaptive space discretization have been proposed (Grüne and Semmler 2004, Grüne 2004). The techniques presented in this work may be categorized as discretization and approximation methods. A discretization of the state space is done, but discretization of the policy space is avoided in order to keep down numerical costs. Instead the optimal controls are found from using first-order conditions in an approximated discrete version of

the Hamilton-Jacobi-Bellman equation. This method relies heavily on the differentiability of the optimal value function, which is satisfied when the object function is concave (see Cotter and Park (2005), and Benveniste and Scheinkman (1979)).

The efficiency of the method depends on the nature of the problem explored. It is, however, well adapted for many problems in resource economics and economics in general, where the object function is concave. Although it does not overcome the dimensionality problem of higher dimensions, it may ease the problem, and a state space of at least four dimensions works fine.

At the end of this work some simple examples with a two-dimensional state space are given, and the relevance of the proposed method is demonstrated and discussed.

3.2 Problem formulation

The general discounted optimal control problem to be solved can be formulated

$$V(x) = \max_{u \in \mathcal{U}} \int_0^\infty e^{-\delta t} g\big(x(t), u(t)\big)\, dt, \tag{3.1}$$

with continuous state equation

$$\frac{d}{dt}x(t) = f\big(x(t), u(t)\big), \qquad x(0) = x_0, \tag{3.2}$$

where $x(t) \in \Omega$ and both $\Omega \subset \mathcal{R}^n$ and $\mathcal{U} \subset \mathcal{R}^m$ are compact sets. Further on, $g(x(t), u(t))$ is continuous and concave with respect to the control variable, $u(t)$.

The solving of this problem requires in many cases numerical procedures. Since numerics is by nature discrete, we may as well give the discrete formulation straight away. By replacing eq. (3.2) with the discrete first order approximation

$$x_h(0) = x, \quad x_h(i+1) = \varphi(x, u) \equiv x_h(i) + f\big(x_h(i), u_h(i)\big)\, h, \tag{3.3}$$

where h is the discrete time-step ($h << 1$), it is shown in Grüne and Semmler (2004) that the corresponding discrete optimal value function is given by

$$V_h(x) \approx \max_{u \in \mathcal{U}} h \sum_{i=0}^{\infty} \beta^i\, g(x_h(i), u_i), \tag{3.4}$$

where $\beta \equiv 1 - \delta h$. Now, if we put eq. (3.3) into eq. (3.4) and take the first period out of the summation sign, we arrive at the discrete Hamilton-Jacobi-Bellman equation

$$V_h(x) \approx \max_{u} \Big\{ h\, g(x, u) + \beta\, V_h\big(\varphi(x, u)\big) \Big\}, \tag{3.5}$$

which is a sufficient condition for optimum.

Solving equation (3.3) repeatedly based on an estimate of $V_h(x)$ is the core of the numerical procedure that will be described and demonstrated in this work. A guess-estimate $V_h^0(x)$ is assumed as a starting value for a fixed-point iteration. Based on this estimate an approximation to $V_h(\varphi(x,u))$ is calculated, and $u^0(x)$ is found as the value that maximizes the right side of the equation based on these estimates. This maximum value, defined as $V_h^1(x)$, is a new and improved estimate of $V_h(x)$, and it leads to a new optimal policy, $u^1(x)$. A sequence of further fixed-point iterations continues until the sequence $\{V_h^1, V_h^2, \dots V_h^n\}$ convergences ($V_h^{n+1}(x) = V_h^n(x)$).

This procedure will in the following be explained mathematically. For notation purposes we start by defining the linear operator

$$L(u)(V_h) \equiv h\, g(x,u) + \beta\, V_h\big(\varphi(x,u)\big). \tag{3.6}$$

The dynamic programming operator

$$T_h(V_h)(x) = \max_{u \in \mathcal{U}} \Big\{ L(u)(V_h) \Big\} \tag{3.7}$$

can be used to successively solve the Hamilton-Jacobi-Bellman equation with fixed-point iterations

$$V_h^{i+1}(x) = T_h(V_h^i)(x), \tag{3.8}$$

and through this a mathematical procedure to solve the original problem stated in eq. (3.1) and (3.2) is established. Since the procedure solves equation (3.5) directly, an optimal solution with many equilibriums is no threat to the procedure. Local versus global extreme points are handled properly as long is this equations is solved. There are, however, many short cuts to make use of, and traps to fall into, when the first steps are taken in concrete numerics.

3.2.1 Numerical procedures

Using first-order conditions in selection of control

To solve the problem defined in eq. (3.1) and (3.2) by numerical methods, we find the optimal value function on a discrete state space grid, $G_x \in \hat{\Omega} \subset \Omega$. This is done by making an initial guess at the value function on G_x and employing the fixed-point iteration in eq. (3.8).

Most literature, e.g. Bertsekas (2001), (2005) and Grüne (2004) deals with working in a discrete "control-space", $\hat{u} = (\hat{u}_1, \hat{u}_2, \dots, \hat{u}_m)$ being a subspace of the continuous value-space $\mathcal{U}$. This should be avoided if possible. First of all the discrete sub-space is sub-optimal unless the problem is in itself discrete with a limited number of nodes or

grid points. Second, depending on the size of the vector $\hat{u}$ and on the dimension of the problem, n, the discretization may be very computer-demanding. Working in $\mathcal{R}^1$ this is no issue, but in $\mathcal{R}^3$ it is very difficult to work with a discrete control vector of some size without struggling with the curse of dimensionality.

One way to cope better with the dimensionality problem is to avoid discretization in control space. Unfortunately, when one is working with a continuous control space, a new problem rises. Assume $x_h(i) = x_i \in \Omega$. Then according to eq. (3.3) admissible controls in a continuous control-space may give a new state variable, $x_h(i+1)$ outside of $\hat{\Omega}$ but inside Ω. The updating of $V_h^{i+1}(x)$ in accordance with (3.8) is therefore not straight-forward. Interpolation in state space, however, is an effective way, which is well documented in literature (see Judd and Solnick (1994)), to find approximate values of the optimal value function for state-values outside the initial selected grid, G_x. In the examples below we use linear interpolation to approximate the value-function outside G_x.

In the selection of optimal control, u, in each fixed-point iteration step that updates the optimal value function according to (3.8), we may take advantage of the first-order Taylor approximation of V_h with respect to x. Inserting

$$V_h\big(\varphi(x,u)\big) \approx V_h(x) + (\nabla V_h)^T(x) f(x,u) h. \tag{3.9}$$

into eq. (3.5), we get

$$V_h(x) \approx \frac{h}{1-\beta} \max_{\underline{u}} \left\{ g(x,u) + \beta(\nabla V_h)^T(x)\, f(x,u) \right\}. \tag{3.10}$$

Only a few values need to be tested to find optimal control, u, that maximize this equation. These are the lower bound $\underline{u} = \underline{0}$, the upper bound and the interior solution(s) solving

$$\frac{\partial g(x,u)}{\partial u} + \beta(\nabla V_h)^T(x)\, \frac{\partial f(x,u)}{\partial u} = 0 \tag{3.11}$$

with respect to u on all the nodes or grid-points of x.

In this equation the only unknown, except from u which is to be found, is $\nabla V_h(x)$. For every fixed-point iteration, however, an estimate for this size can be found. Typically, we start the fixed-point iteration with $V_h^0 = 0$ and use the nodes or grid points of x to estimate $\nabla V_h^k(x)$ from

$$\nabla V_h^k(x) \approx \frac{V_h^{k-1}(x+\Delta x) - V_h^{k-1}(x)}{\Delta x}. \tag{3.12}$$

Then $V_h^k(x)$ will, for each fixed-point iteration, approach its true value, $V_h(x)$, and $u^k(x)$ will approach $u_h(x)$.

3.2.2 Policy and value iterations

The number of heavy approximations should be reduced to a minimum in order to improve time of convergence. One way to do that is to mix policy iterations and what we call "value iterations" (see Grüne and Semmler (2004)).

In the solving procedure of the Hamilton-Jacobi-Bellman equation (3.5), each step of using the dynamic programming operator (3.7) could be called a policy iteration. To speed up convergence switching between this policy-iteration and less numerically expensive value iterations can be done. In the "value-iterations" the policy is fixed, $u = u^f$, and therefore also the profit, $g(x, u^f)$, is fixed in the linear value iteration operator (3.6). This means that the value iterations

$$V_h^{n+1} = L(u^f)V_h^n \tag{3.13}$$

stabilize rather quick. When they do, we shift to policy-iterations in accordance with equation (3.7), before we again return to value-iterations. The alternation between value and policy iterations continues until convergence.

3.2.3 Interpolating first-order conditions

The usefulness of utilizing first order conditions in accordance with the procedure described in sec. 3.2.1 depends on the numeric costs of solving equation (3.11) with respect to u. When exact roots can be found algebraically the method is very favorable, but it might be preferable also in cases where the numerics are more challenging.

For many cases an exact root of (3.11) is not required. When approximate solutions are sufficient, a passable and effective procedure is to find unique interpolated functions on each node of G_x representing the u-dependence. Say, for node i on G_x we find $h_1(i, u)$ and/or $h_2(i, u)$ that satisfies $h_1(i, u) \approx \frac{\partial g(x,u)}{\partial u}$ and/or $h_2(i, u) \approx \frac{\partial f(x,u)}{\partial u}$. If these interpolations are sufficiently good approximations and have forms that make it possible algebraically to find roots of

$$h_1(i, u) + \beta(\nabla V_h)^T h_2(i, u) = 0, \tag{3.14}$$

the method can be used. (Eq. (3.14) corresponds to eq. (3.11).

Note that interpolating $f(x, u)$ and $g(x, u)$, and taking the partial derivative of these functions instead of interpolating the partial derivatives directly, may lead to large errors.

3.3 Examples

In this section we will look at some examples from two classes of problems where the first-order conditions in eq. (3.11) are effective in the finding of optimal control. The

first class of problems consists of cases where an algebraic solution is reachable (examples 3.3.1 and 3.3.2), and the second class is problems where an algebraic solution is reachable only when $\frac{\partial f(x,u)}{\partial u}$ and/or $\frac{\partial g(x,u)}{\partial u}$ is replaced by simpler functions on G_x (see example 3.3.3). Such a replacement should only be done when the interpolated functions are sufficiently close to the original ones.

3.3.1 Example 1: A 2d investment model with relative adjustment cost

First we will look at a problem where eq. 3.11 is easily solved algebraically without any interpolation. The problem from Haunschmied and colleagues (2003) is also presented in Grüne and Semmler (2004), where it is solved by adaptive grid schemes. The inputs to equation (3.1) and (3.2) are respectively given by

$$g(x,u) = k_1\sqrt{x_1} - \frac{x_1}{1+k_2x_1^4} - c_1x_2 - \frac{c_2x_2^2}{2} - \frac{\alpha u^2}{2}, \tag{3.15}$$

and

$$f(x,u) = \begin{pmatrix} x_2(t) - \sigma x_1(t) \\ u(t) \end{pmatrix}. \tag{3.16}$$

The state x_1 is capital stock, x_2 is investment, and the control, u, should be interpreted as change in investment. The parameters are $k_1 = 2, \quad k_2 = 0.0117, \quad c_1 = 0.75, \quad c_2 = 2.5, \quad \alpha = 12, \quad \sigma = 0.25$, and discount rate $\delta = 0.04$. Subsequently, we choose time-step $h = 1/20$.

The procedure is rather easy, but let us go through the first example step by step to convince all readers.

1. Let G_x be the 2D grid with $x_1 = \{0,\ 0.05,\ 0.1,\ ...,\ 10\}$ and $x_2 = \{0,\ 0.01,\ 0.02,\ ...,\ 3\}$ and let $h = 1/20$. This means that G_x is a very fine-meshed grid.

2. Start with a guess estimate for $V_h(x)$, e.g a zero-matrix on the whole x-grid, G_x.

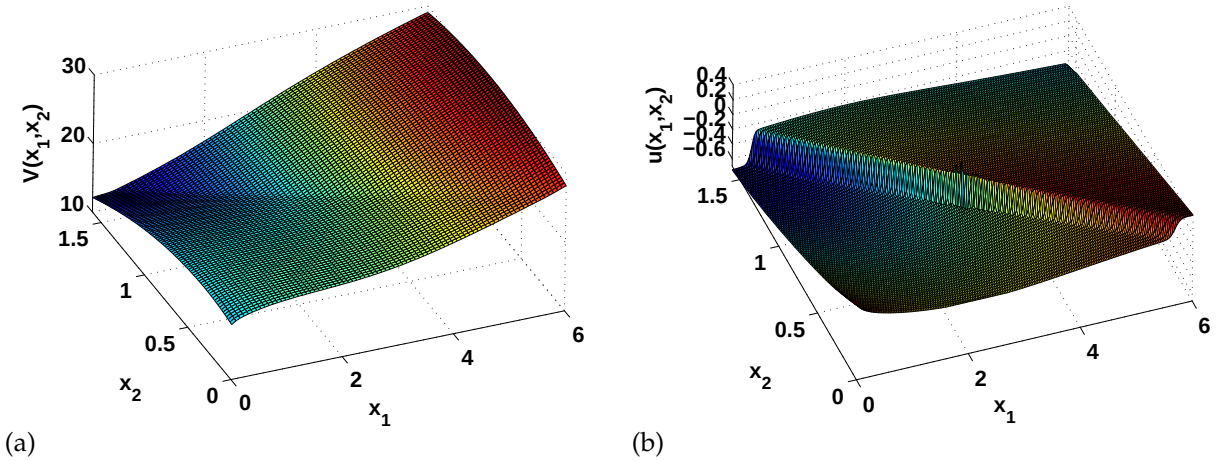
3. Find $\nabla V_h(x)$ with equation (3.12). If the zero-matrix is chosen as a start value for $V(x)$, the first $\nabla V_h(x)$ will also be the zero-matrix.

4. On G_x, find the set of controls $u(x)$ that maximizes $V(x)$ according to equation (3.11). Since $g(x,u)$ is concave with respect to u the inner solution will be the maximum. Putting $\partial g/\partial u = -\alpha\, u$ into the equation gives $-\alpha\, u + \beta \nabla V_h(x)\, \partial f/\partial u = 0$, and consequently the algebraic solution is $u(x) = \beta\, \nabla V_h(x)/\alpha$.

5. Update $V_h(x)$ according to 3.11

6. Start again from point 1 with the updatet $V_h^1(x)$ as initial guess and repete the whole procedure until convergence

A matlab code that solves the problem numerically is given in appendix B. In the code one should regard that the finding of optimal policy according to equation (3.14) is done by a complete matrix operation. That is, instead of using if- and for-loops through the grid, G_x, the calculation of $u(x)$ is done on the whole G_x in a single operation.



(a) (b)

Figure 3.1: Example 1: (a) the optimal value function and (b) the optimal control on the state space grid.

Figure 3.1(b) shows a special characteristic of the optimal control. There are discontinuities in the politics along a line which, in literature, is referred to as a skibaline or a DNS line. (See Haunschmied 2003). The discontinuity slows down convergence, and for these kinds of problems adaptive grid schemes are ideal for treating the behavior of the optimal control. Utilization of the first-order conditions, however, allows selection of a fine-meshed grid G_x, and therefore we did not need an adaptive grid to find a satisfactory solution. (See Figures 3.1(a) and 3.1(b).)

3.3.2 Example 2: Optimal harvest of two species

The following example concerns management of a two-species fish resource. One of the species predates the other, and both of the species are harvested. Therefore this model is two-dimensional in both control and state space, and a discretization of the control space would have been even more numerically expensive than that for example 1.

The profit and dynamics to be putted into equation (3.1) and (3.2) respectively are

$$g(x,u) = \sum_{i=1}^{2} p_i\, u_i - \frac{c_i}{x_i}\, u^{\alpha_i} \tag{3.17}$$

and

$$f(x,u) = \begin{pmatrix} r_1 x_1 \left(1 - \frac{x_1}{k_1} - b_1 x_2\right) \\ r_2 x_2 \left(1 - \frac{x_1}{k_2} + b_2 x_1\right) \end{pmatrix}, \tag{3.18}$$

and the constants are $r_1 = r_2 = 1$, $p_1 = 1$, $p_2 = 25$, $c1 = c2 = 0.1$ $\alpha_1 = 2$, $\alpha_2 = 2.1$, $k_1 = k_2 = 1$, $b_1 = 0.001$, $b_2 = 0.0001$ and the discount rate $\delta = 0.05$.

Optimal controls, catch u_1 and u_2, are found directly from equation 3.11, and are given algebraically by

$$u_i = \left(\frac{x_i}{2}\left(p_i - \beta\frac{\partial V}{\partial x_i}\right)\right)^{1/(\alpha_i - 1)}. \tag{3.19}$$

Convergence to the optimal curves presented in Figures 3.2(a) and 3.2(b) is reached within half a minute when a x-grid, G_x of size 44×44, and time-step $h = 1/1000$ was chosen.

3.3.3 Example 3: Interpolation in the first-order conditions

The dynamics in this example is identical with that in example 3.3.2, but the object function is slightly changed to

$$g(x,u) = p_1\, u_1 - \frac{c_1}{x_1}\, u^{\alpha_1} + (p_2 - p_3 u_2)u_2 - \frac{c_2}{x_2}\, u_2^{\alpha_3}. \tag{3.20}$$

The constants have the same values as in example 3.3.2, but $p_3 = 1$ and $\alpha_3 = 1.1$ are new to this problem.

The change of object function has consequences for the solving of equation 3.11, which is no longer analytically solvable for u_2. A direct numerical approach leads to very high throughput, and therefore we should search for a short path omitting that problem. Very often it is possible to find simple interpolated functions that are sufficiently close to $g(x,u)$ to be substituted into equation (3.11) (as in equaton (3.14)). These should be functions that make it possible to solve the equation either analytically or with fast numerical algorithms.

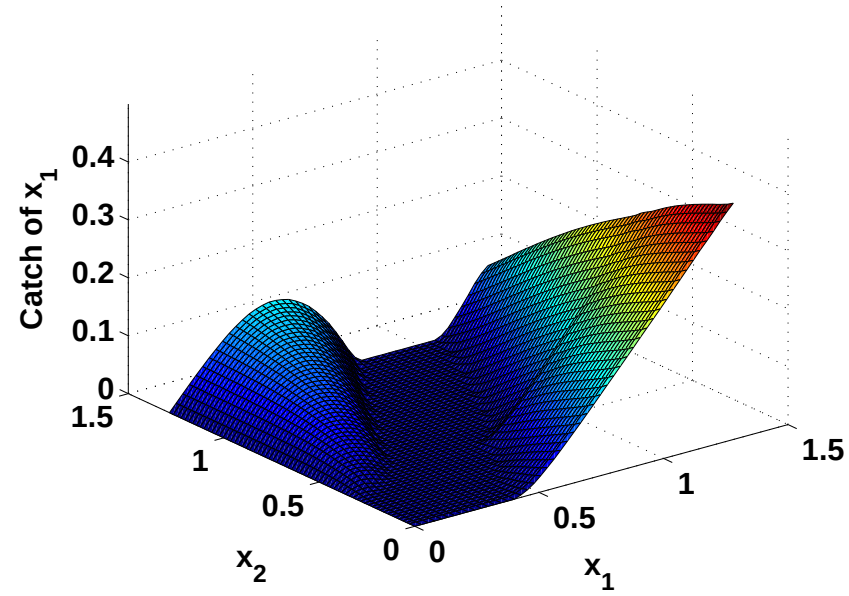
In this problem

$$\frac{\partial g(x,u)}{\partial u_2} = 25 - 2u_2 - \frac{0.11}{x_2} u_2^{0.1}. \tag{3.21}$$

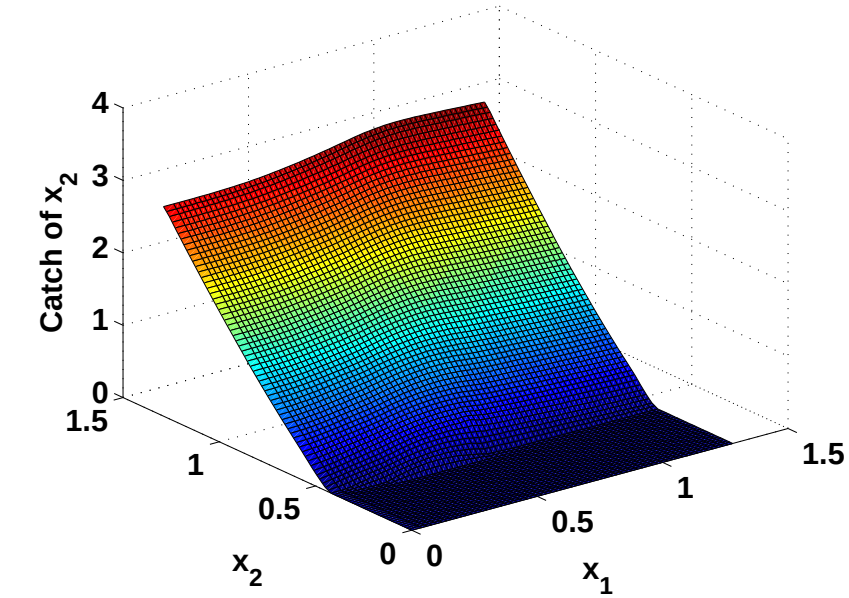
This suggests that the linear function

$$h_1(i,u_2) = A(x(i)) + B(x(i))u_2 \tag{3.22}$$

might be a sufficiently good approximation for $\frac{\partial g(x,u)}{\partial u_2}$. We have to test the quality, however. A graphical inspection shows that there is practically no difference between $\frac{\partial g(x,u)}{\partial u_2}$ and the interpolated function (see Figure 3.3.3). Only close to $x_2 = 0$ are there some



(a)



(b)

Figure 3.2: Example 2: The optimal catch of (a) species x_1 and (b) species x_2 on the state space grid G_x.

differences owing to extreme growth of the last term of eq. (3.21) when x_2 approaches zero. The lowest x_2-value in Figure 3.3.3 is 0.01.

When $\frac{\partial g(x,u)}{\partial u_2}$ is replaced with $x(i) + B(x(i))u_2$ in eq. (3.11), optimal catch is given by

$$u_2 = \left(\beta \frac{\partial V_h(x)}{\partial x_2} - A(x(i))\right) \Big/ B(x(i)). \tag{3.23}$$

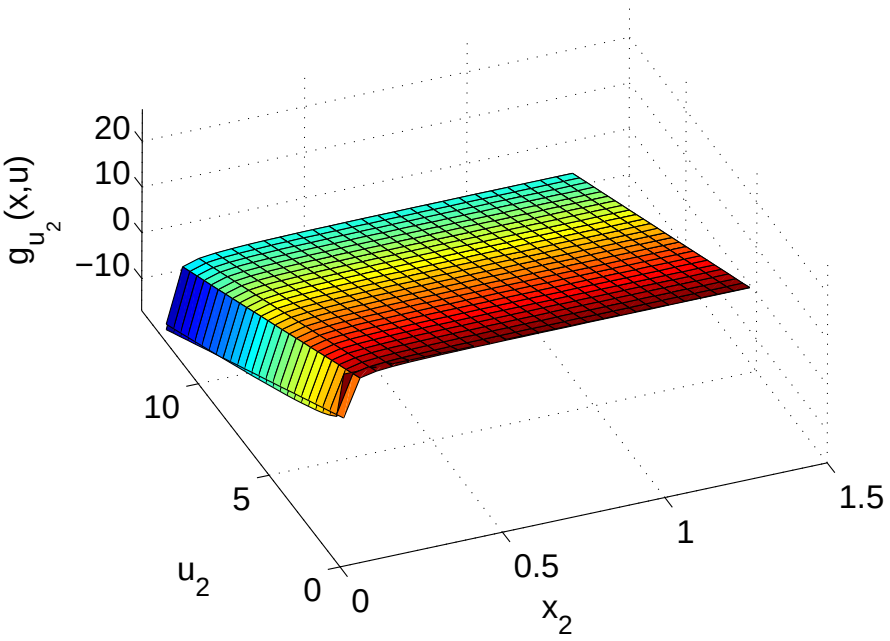


Figure 3.3: Example 3: The surfaces of $\frac{\partial g(x,u)}{\partial u_2}$ and the interpolated replacement function $h_1(i, u_2) = A(i) + B(i)u_2$ plotted in the same figure.

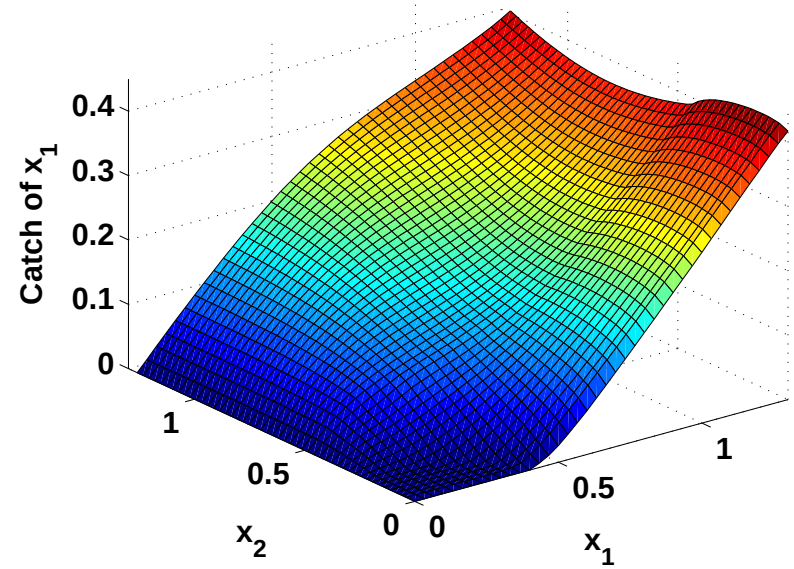
The optimal catches of the two species are plotted in Figures 3.4(a) and 3.4(b).

3.4 Range of use for the numerical procedure

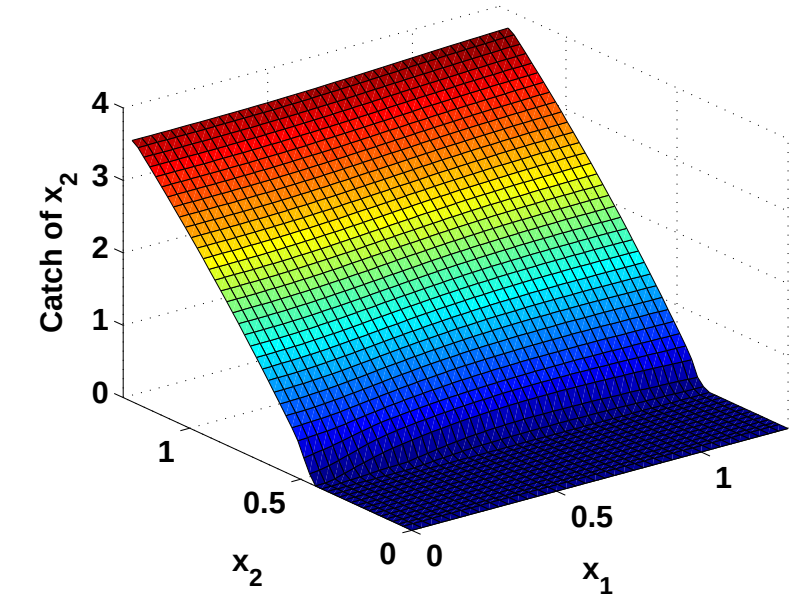
The numerical procedure described in section 3.2.1 is not always effective. It is a necessary condition that there exists a method to find all solutions to equation (3.11) algebraically or numerically within a limited number of iterations.

For resource management models, where u typically is the harvest or the use of a certain resource, the function $f(x, u)$ is usually linear with respect to u. This simplifies equation (3.11) as it is reduced to

$$\frac{\partial g(x,u)}{\partial u} + \beta(\nabla V_h)^T(x) \cdot K = 0, \tag{3.24}$$



(a)



(b)

Figure 3.4: Example 3: The optimal catch of (a) species x_1 and (b)species x_2 on the state space grid G_x.

where K is some constant, usually $K = 1$. We will not go into any discussions about numerical solvability of such equations since it on its own is a special field written tons of

literature about. However, when either $g(x,u)$ or $\partial g(x,u)/\partial u$ is quadratic, which often is the case in social economic and bioeconomic literature, it is simple to solve equation (3.24) algebraically. When that is the case, as in example 1 and example 2, the method is preferable. It should also be considered when (3.24) can be solved numerically within a reasonable number of iterations or when an approximation to this equation can be solved either algebraically, as in example 3, or numerically. This means that the method is very flexible for numerically experienced users.

How to numerically solve equation (3.10) effectively is a question that should be paid attention. In the problem of *example 1* the solution is characterized by a discontinuity in the control space, and it is already mentioned that an adaptive grid method is effective in that case. Since adaptive grids may easily be used in the solving of equation (3.10) our procedure should be combined with adaptive grids for problems of this kind. With an adaptive grid G_x is rather crude in the beginning but, depending on the sizes of local changes in each of the fixed-point iterations, the grid is refined around the nodes where the changes are largest. Adaptive grid is, however, not possible to implement without breaking up the matrix operations on G_x when finding optimal controls, u. Instead of finding $u(x)$ in a single matrix-operation (see matlab-code in appendix B), it is necessary to go through each element (node) of the matrixes by if- and for-loops. Although that increases the time to find each $u(x)$ on $G(x)$, the total time to solve the problem may decrease since the adaptive grid for most cases allows a lot cruder G_x (less nodes on G_x). Especially, this is the case when the optimal control, $u(x)$, is discontinuous.

3.5 Conclusions

In this work we have demonstrated the use of some efficiency-improving methods for the solving of optimal control problems with dynamic programming. These methods do not overcome the curse of dimensionality, but their efficiency in making dynamic programming solutions feasible and attractive for many problems with state space of up to four dimensions is unquestionable.

In the methods demonstrated, discretization in state space is employed (discretization methods). Subsequently, a combination of Taylor approximation of the optimal value function and first-order conditions with respect to the optimal controls is used to decide controls in accordance with the discrete Hamilton-Jacobi-Bellman equation. When an optimal control is not analytically solvable in the discrete two-stage problem, we may use interpolation and approximation techniques to find analytic solutions to related approximated problems that are good representations of the original problem.

3.6 References

Bellman, R. 1957. "Dynamic programming". Princeton University Press. Princeton, New Jersey.

Benveniste, L.M. and J. A. Scheinkman. 1979. "On the differentiability of the value function in dynamic models of economics". Econometrica 47:727-732

Bertsekas, D.P. 2001. "Dynamic Programming and Optimal Control", vol. 1, 3rd ed. Athena Scientific. Belmont, Massachusetts.

Bertsekas, D.P. 2005. "Dynamic Programming and Optimal Control", vol.2, 2nd ed. Athena Scientific. Belmont, Massachusetts.

Bertsekas, D.P. and J. Tsitsiklis. 1995. "Neuro-dynamic programming: an overview". In Proceedings of the 34th Conference on Decision and Control, vol. 1, pp. 560-564. Springer, Berlin Heidelberg.

Cotter, K.D. and J.H. Park. (2005). "Non-concave dynamic programming". Economic Letters 90:141-146.

Grüne, L. 2004. "Error estimation and adaptive discretization for the discrete stochastic Hamilton-Jacobi-Bellman equation". Numerische Mathematik 99:85-112.

Grüne, L. and W. Semmler. (2004). "Using Dynamic Programming with Adaptive Grid Schemes for Optimal Control Problems in Economics". Journal of Economic Dynamics & Control. 28:2427-2456.

Haunschmied,J.L., Hort, P.M., Hartl, R.F. and G. Feichtinger. 2003. "A DNS-curve in a two-state capital accumulation problem: A numerical analysis". Journal of Economic Dynamics and Control. 27:701-716

Judd, K.L., and A. Solnick. 1994. "Numerical dynamic programming with shape-preserving splines". Mimeo. Available at http://bucky.stanford.edu/papers/dpshape.pdf.

Rust, J. 1996. "Numerical Dynamic Programming". Pp. 620-729 in H.M. Amman, D.A. Kendrick and J. Rust (eds.): Handbook of Computational Economics, 1st ed. Elsevier. North Holland.