Solving Optimal Switching Models

Paul L. Fackler*

June 26, 2004

Abstract

Numerous control problems in economics can be characterized as optimal switching problems, including optimal stopping and entry/exit problems. With certain notable exceptions, solutions to such problems must be computed numerically. This paper discusses an approach to obtaining a numerical solution in a fairly general setting and describes a user-friendly MATLAB implementation that simplifies that process.

Keywords: Optimal switching, stochastic control,computational methods

JEL classification codes: C61, C63, C88

*The author is an Associate Professor at North Carolina State University.

Mail: Department of Agricultural and Resource Economics

NCSU, Box 8109

Raleigh NC, 27695, USA

e-mail: `paul_fackler@ncsu.edu`

Web-site: `http://www4.ncsu.edu/~pfackler/`

Solving Optimal Switching Models

Abstract

Numerous control problems in economics can be characterized as optimal switching problems, including optimal stopping and entry/exit problems. With certain notable exceptions, solutions to such problems must be computed numerically. This paper discusses an approach to obtaining a numerical solution in a fairly general setting and describes a user-friendly MATLAB implementation that simplifies that process.

Keywords: Optimal switching, stochastic control, computational methods

JEL classification codes: C61, C63, C88

Many problems in stochastic control involve situations in which an agent can choose among a discrete set of states or regimes. The choice of a control can be thought of as the choice of the regime. In addition, a diffusion process drives one or more state variables that affect the returns associated with each of the regimes. The total returns to the agent are also affected by the existence of costs associated with switching among the regimes.

Optimal switching models often arise in pricing so-called real options. The recent collections of Brennan and Trigeorgis (2000) and Schwartz and Trigeorgis (2001) provide an introduction to this area and numerous examples. The widely read text by Dixit and Pindyck (1994) also contains numerous examples of optimal switching models. The real options literature recognizes and attempts to value such things as the flexibility to defer action, to change from one activity to another, to abandon an investment or to default on a project.

The simplest optimal switching models are optimal stopping problems, where one of the regimes represents continuation and other represents a permanent "stopped" state. The American option pricing problem is a well known example. More complicated optimal switching models allow for movement back and forth among regimes and may have more than two regimes. A well known example introduced in McDonald and Siegel (1985), McDonald and Siegel (1986), and Brennan and Schwartz (1985), is that of firm entry/exit. In such problems, a firm can make an investment in an asset that produces an output with a stochastic price. When the price is high enough, it is worth activation though investment in the asset. When the price is low enough, however, the firm may abandon its investment and become inactive. In the basic entry/exit model there are two regimes, active and inactive. An extension of this model adds a third regime that allows for temporary suspension of production.

Brekke and Oksendal (1994) have provided general solution conditions for optimal switching models that can be expressed as a set of functional complementarity conditions. In most cases explicit solutions to these conditions cannot be obtained. This paper discusses how numerical solutions can be obtained using function approximation and collocation. The approach results in a type of problem known as an Extended Vertical Linear Complementarity Problem (EVLCP) for which a number of solution algorithms exist. A user friendly interface for defining and solving optimal switching models using MATLAB is documented and illustrated.

Problem Statement and Optimality Conditions

The general optimal switching model applies to problems in which an agent must choose, at each moment, among m regimes, $R \in \{1, \ldots, m\}$. The agent can move from regime i to j at a cost of C_{ij} (which may be arbitrarily high, thereby ruling out such a movement). C represents a lump-sum switching cost with $C_{ii} = 0$, i.e., there is no lump sum cost to remaining in the current regime. The agent also receives a flow of payments per unit time of $f(S, R)$, which depends on both the active regime and on a continuous d-dimensional state process S that is described by

$$dS = \mu(S, R)dt + \sigma(S, R)dW. \tag{1}$$

The agent desires to maximize over an infinite time horizon the discounted value (discounted at rate $\rho(S)$) of the flow of payments received less any switching costs incurred.

Brekke and Oksendal (Theorem 3.4) have shown that the optimal value function $V(S, R)$ satisfies[1]

$$\rho(S)V(S, R) \geq f(S, R) + \mu(S, R)V_S(S, R) + \frac{\sigma^2(S, R)}{2}V_{SS}(S, R) \tag{2}$$

and the $m - 1$ conditions

$$V(S, R) \geq V(S, x) - C_{Rx}, \ \ \forall x \neq R \tag{3}$$

Furthermore, one of these m conditions must be satisfied with equality at each (S, R). Which one is satisfied with equality determines the optimal policy. Thus, if $V(S, R) = V(S, x) - C_{Rx}$, for some x, it is optimal at S to switch from R to x. Otherwise it is optimal to remain in regime R and the first condition is satisfied with equality.

There is a simple economic intuition behind these conditions. The value function can be thought of as the value of the assets that generate the payment flows. By Ito's Lemma the

[1]If S is multidimensional, this condition is more accurately written as

$$\rho V \geq f + \sum_i \mu_i \frac{\partial V}{\partial S_i} + \frac{1}{2}\sum_i \sum_j \Sigma_{ij} \frac{\partial^2 V}{\partial S_i \partial S_j}$$

where $\Sigma_{ij} = \sum_k \sigma_{ik}\sigma_{jk}$. The simpler form is to avoid notational clutter.

expected rate of appreciation of those assets is

$$\frac{dE[V(S,R)]}{dt} = \mu(S,R)V_S(S,i) + \frac{\sigma^2(S,R)}{2}V_{SS}(S,R). \tag{4}$$

The total rate of return when regime R is active equals the current return flow $f(S,R)$ plus the expected rate of capital appreciation $dE[V(S,R)]/dt$. Thus, the first condition states that the rate of return obtainable by investing V dollars must be at least as great as the total rate of return generated by the assets if one remains in regime R. The second condition states that the value function must be at least as great as the value that could be obtained by switching regimes.

Anticipating the numerical approach discussed in the next section, the optimality conditions can be written in the following equivalent form

$$0 = \min\left(\beta(S,R) - f(S,R), \min_{x\neq R} V(S,R) - V(S,x) + C_{Rx}\right) \tag{5}$$

where

$$\beta(S,R) = \rho(S)V(S,R) - \mu(S,R)V_S(S,R) - \frac{\sigma^2(S,R)}{2}V_{SS}(S,R) \tag{6}$$

It should also be pointed out that this condition does not fully characterize the solution. In particular, additional regularity conditions are needed to uniquely define V. Essentially these amount to conditions that rule out explosive growth in the value function.

To illustrate the framework, an example from Brekke and Oksendal is reviewed. Consider a mine currently containing Q units of ore. The mine is either idle ($R = 1$) or ore is extracted at rate hQ ($R = 2$) with a fixed cost of k incurred. The transition equation for Q is thus

$$dQ = \begin{cases} 0 & \text{if } R = 1 \\ -hQdt & \text{if } R = 2 \end{cases} \tag{7}$$

The current price at which ore can be sold evolves according to a geometric Brownian motion

$$dP = \mu P dt + \sigma P dW \tag{8}$$

The flow of returns to the mine is

$$f(Q,P,R) = \begin{cases} 0 & \text{if } R = 1 \\ hQP - k & \text{if } R = 2 \end{cases} \tag{9}$$

The firm incurs fixed startup and shutdown costs of C_{12} and C_{21} and uses a fixed discount rate of ρ.

The solution conditions are

$$\begin{aligned} 0 = \min\Big(&\rho V(Q,P,1) - \mu P V_p(Q,P,1) - \tfrac{1}{2}\sigma^2 P^2 V_{PP}(Q,P,1), \\ &V(Q,P,1) - V(Q,P,2) + C_{12}\Big) \end{aligned} \tag{10}$$

and

$$\begin{aligned} 0 = \min\Big(&\rho V(Q,P,2) - \mu P V_p(Q,P,2) - \tfrac{1}{2}\sigma^2 P^2 V_{PP}(Q,P,2) \\ &+ hQV_Q(Q,P,2) - (hQP - k),\ V(Q,P,2) - V(Q,P,1) + C_{21}\Big) \end{aligned} \tag{11}$$

As Brekke and Oksendal point out, the problem can be simplified by defining $y = QP$ and noting that

$$dy = \begin{cases} \mu y dt + \sigma y dW & \text{if } R = 1 \\ (\mu - h) y dt + \sigma y dW & \text{if } R = 2 \end{cases} \tag{12}$$

and

$$f(Q,P,R) = f(y,R) = \begin{cases} 0 & \text{if } R = 1 \\ hy - k & \text{if } R = 2 \end{cases} \tag{13}$$

Expressing the value function in terms of y and R, the optimality conditions are

$$0 = \min\Big(\rho V(y,1) - \mu y V_y(y,1) - \frac{1}{2}\sigma^2 y^2 V_{yy}(y,1),\ V(y,1) - V(y,2) + C_{12}\Big) \tag{14}$$

and

$$0=\min\Big(\rho V(y,2)-(\mu-h)yV_y(y,2)-\tfrac{1}{2}\sigma^2y^2V_{yy}(y,2)-(hy-k),\\ V(y,2)-V(y,1)+C_{21}\Big) \tag{15}$$

$\forall y\in[0,\infty)$.

Numerical Solutions Methods

Optimal switching models generally require numerical approximations. In simple models the functional form of the solution may be known and numerical methods are required only to compute a limited set of parameter values. In particular, problems with a single geometric Brownian motion state, such as the one-dimensional formulation of the example given in the previous section, often can be solved in this way (see Dixit and Pindyck (1994) for more examples and nearly explicit solutions).

For general problems, however, the functional form of the solution may not be known. Projection methods (Judd (1998), Miranda and Fackler (2002)) using complete families of approximating functions represent a natural way to find approximate solutions to such models. Suppose that $V(S,i)$ is approximated by $\phi(S)\theta_i$, where ϕ represents a set of n basis functions for a family of approximating functions and θ_i is an n-vector of coefficients for the value function associated with the ith regime. Define the approximate differential operator

$$\beta(S,i)=\rho(S)\phi(S)-\mu(S,i)\phi'(S)+\frac{\sigma^2(S,i)}{2}\phi''(S) \tag{16}$$

The inequality conditions can now be written as

$$\beta(S,i)\theta_i-f(S,i)\geq 0 \tag{17}$$

and

$$\phi(S)\theta_i - \phi(S)\theta_j + C_{ij}(S) \geq 0, \;\; \forall j \neq i \tag{18}$$

Values of the θ_i can be obtained by collocation, which solves the optimality conditions at a set of n nodal state values $\{s_k\}$. Define the $n \times n$ matrices Φ and B_i as the functions $\phi(S)$ and $\beta(S, i)$ evaluated at the nodal values. Similarly, define f_i to be the n-vector of values of $f(S, i)$ evaluated at the n nodal state values.

The problem can now be stated as an extended vertical linear complementarity problem (EVLCP),[2] which seeks a solution z to

$$0 = \min(M_1 z + q_1, M_2 z + q_2, \ldots, M_m z + q_m). \tag{19}$$

where the M_i are each $N \times N$ and the q_i are each $N \times 1$ and where the min operator is applied element-wise. The EVLCP could also be written as $w_i = M_i z + q_i \geq 0$ for $i = 1, \ldots, m$ and

$$\prod_{i=1}^{m} w_i = 0 \tag{20}$$

where the product is taken element-wise. The solution to an EVLCP problem thus requires that each of the w_i be nonnegative and that for each of the N elements, at least one of the m w_i values is exactly zero. Unlike more common complementarity conditions (such as the Karush-Kuhn-Tucker conditions) the complementarity here extends over m variables rather than only over two and hence cannot be written as a vector orthogonality condition.

In the general form z represents the m column vectors θ_i stacked vertically. The M_i are given by

$$M_i = e_i {e_i}^\top \otimes B_i + (I_m - \underline{1}_m {e_i}^\top) \otimes \Phi \tag{21}$$

[2]This generalization of the standard linear complementarity problem has a number of names in the literature including the extended generalized order LCP (Gowda and Sznajder (1994)).

and the q_i are given by

$$q_i = \begin{bmatrix} C_{1i}\underline{1}_n \\ \dots \\ C_{mi}\underline{1}_n \end{bmatrix} - [e_i \otimes f_i] \tag{22}$$

where $\underline{1}_m$ is a column vector composed of m ones and e_i is the ith column of an $m \times m$ identity matrix. Thus

$$M_1 = \begin{bmatrix} B_1 & 0 & \cdots & 0 \\ -\Phi & \Phi & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ -\Phi & 0 & \cdots & \Phi \end{bmatrix}, \quad q_1 = \begin{bmatrix} -f_1 \\ C_{21}\underline{1}_n \\ \cdots \\ C_{m1}\underline{1}_n \end{bmatrix} \tag{23}$$

$$M_2 = \begin{bmatrix} \Phi & -\Phi & \cdots & 0 \\ 0 & B_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & -\Phi & \cdots & \Phi \end{bmatrix}, \quad q_2 = \begin{bmatrix} C_{12}\underline{1}_n \\ -f_2 \\ \cdots \\ C_{m2}\underline{1}_n \end{bmatrix} \tag{24}$$

etc.

A number of solution approaches for solving EVLCPs have been proposed, each based on solution approaches that have proven useful for related problems. Cottle and Dantzig (1970) proposed a pivoting strategy for a related problem that is based on Lemke's algorithm for solving ordinary LCPs (Lemke (1965)).[3] Details on modifications to the Lemke-based algorithm

[3]Cottle and Dantzig (1970) proposed this algorithm for what they termed the generalized LCP, which has since become known as the vertical LCP. This is a special case of the EVLCP in which one of the M_i is an identity matrix and the associated q_i is a vector of zeros.

for EVLCPs are available from the author. In the first discussion of the application of EVLCPs to control theory, Sun (1989) proposed an iterative algorithm similar in spirit to the Projected Successive Over-Relaxation (PSOR) method for solving LCPs. This method, however, can be unstable for solving the kind of problems considered here. Recently, a smoothing Newton method using a so-called aggregation function (also known as an entropy function) has been proposed by Qi and Liao (1999). In numerical trials for the kind of problems considered here, the smoothing Newton method has the best performance characteristics.

To date theoretical results on the solvability of EVLCPs are mostly limited to the case in which all row-representative matrices associated with the M_i are nonsingular with determinants of the same sign, the so-called W property (see Gowda and Sznajder (1994)). Unfortunately, in the present case, singularity of a row representative matrix can result, and hence these results do not apply. It also means that it is possible for the Lemke-based and smoothing Newton based methods to fail. In practice, however, this does not seem to be a problem.

The quality of the approximate solution depends on both the choice of the family of approximating functions and the set of nodal values used to obtain the collation solution. In general $V(S,R)$ is not smooth, but exhibits discontinuity in the second derivative at points for which it is optimal to switch from R to another regime.[4] For this reason polynomial approximations generally perform poorly. A better alternative is to use either piecewise linear functions (using finite differences to approximate ϕ' and ϕ'') or cubic spline functions because these functions are not as adversely affected by the derivative discontinuities.

[4]If $C_{ij} = C_{ji} = 0$ at a point S for which is it optimal to switch between i and j, then the value function will exhibit discontinuity in the third derivative (see the related discussion of super-contact conditions in Dumas (1991))

A MATLAB Implementation

To specify an optimal switching model, an analyst must define the functions $f(S, R)$, $\rho(S)$, $\mu(S, R)$ and $\sigma(S, R)$ and assign values to any parameters that these function use as well as to the cost parameter matrix C. To solve the model using the function approximation approach described in the previous section also requires that the analyst specify the family of approximating functions to be used and the nodal values of S at which to evaluate the complementarity conditions.

This section describes an implementation in MATLAB that makes the solution process relatively simple (the code described here can be downloaded from the author's website). The first step is to code a model function file according to the following template:

```
function out=func(flag,S,R,additional parameters)
switch flag
case 'f'
  out = reward function f(S,R);
case 'mu'
  out = drift function mu(S,R);
case 'sigma'
  out = diffusion function sigma(S,R);
case 'rho'
  out = discount rate rho(S);
end
```

The procedure will be passed a string variable `flag`, an $nm \times d$ matrix of values of the continuous state S, a scalar value of the regime R and any additional parameters needed to evaluate the model functions. When the `flag` variable is `f` the procedure should return an $nm \times 1$ vector, when `flag` is `mu` it should return an $nm \times d$ matrix, when `flag` is `sigma` it should return an $nm \times d \times d$ array and when `flag` is `rho` it should return an $nm \times 1$ vector (or a scalar if ρ is a constant).

The model specification is completed by defining a structure variable `model` with the following three fields:[5]

`func`	the name of the model function file
`params`	a cell array with the parameters to pass to the model function file
`cost`	the $m \times m$ switching cost matrix C

The family of approximating functions is specified by using the procedures in the CompEcon Toolbox described in Miranda and Fackler (2002) which is available from Fackler's website. The toolbox procedures can be used to create a structure variable called `fspace` containing all the information needed to define the necessary basis matrices.

The main solution procedure is a procedure called `ossolve` which has the following syntax:

```
[cv,snodes,x,xindex]=ossolve(model,fspace,snodes,cv);
```

The first two inputs have already been described and are the only ones needed. The third input, if passed, is a set of nodal values of S. If S is one-dimensional, this should be a simple column vector. If S is d-dimensional, `snodes` should be passed as a cell array of column vectors. These will be expanded by `ossolve` into a grid of values, the size of which, n, will equal the product of the lengths of the vectors. The fourth input, if passed, is an $n \times m$ matrix of initial values for the coefficients of the value function approximations.

The first output returned by the procedure is an $n \times m$ matrix of coefficients of the value function approximations. The value function can now be evaluated at arbitrary values of (S, R) using the CompEcon Toolbox function call

[5]Structure variables in MATLAB are user defined data type with named fields. The data in a field is accessed using the syntax `variablename.fieldname`. Cell arrays are data types with fields accessed by index number, e.g., `variablename{i,j}`.

```
V=funeval(cv(:,R),fspace,S);
```

The marginal value (shadow price) function can be obtained in a similar fashion using

```
dV=funeval(cv(:,R),fspace,S,1);
```

The second and third outputs provide the nodal values used in finding the solution ($nm \times d$) and the optimal regime choice at the nodal values ($nm \times m$). This provides a representation of optimal decision rule. This representation, of course, is only as accurate the mesh size of the grid of nodal points.

The fourth output `xindex` ($m \times 6$) is useful for characterizing on the optimal decision rule in one dimensional problems. For each of the m regimes it contains information about the lower and upper boundaries of the no-switch region. The first column contains the approximate location of the lower bound, the second column contains the regime number that it is optimal to switch to at that point and the third column contains the difference in the derivatives of the value functions for the two regimes at the approximate switch point (this difference should be zero and thus provides a check on the solution). Columns 4 through 6 provides similar information at the upper bound of the no-switch region. `xindex` is returned as an empty matrix when $d > 1$.

A separate utility is also provided to evaluate the optimal decision rule and the value function at arbitrary points and works for any value of d. It is called with the following syntax:

```
[regime,V,dV]=osoptimum(S,R,cv,model,fspace);
```

The first input is a $k \times d$ matrix of arbitrary values of S. The second input R is either scalar or a $k \times 1$ vector of values of the regime number. The remaining inputs are the coefficient matrix returned from `ossolve` and the model definition and function definition structures passed to `ossolve`.

The utility returns a best guess of the optimal decision rule, the value function and the marginal value (shadow price) function at the points (S, R) (the sizes of the outputs are $k \times 1$, $k \times 1$ and $k \times d$, respectively). The optimal decision rule is determined by calculating the value function at (S, R) as well as the value function at S for the other regimes less the cost of switching to them from R. The best switch is computed and then compared to the no-switch strategy.

It is, however, impossible to determine with complete satisfaction the location of no-switch boundary due to the approximations inherent in the solution approach. The value function for any given regime should be exactly equal to the value of the best switch strategy outside the no-switch region and should be strictly greater everywhere within the region. As a compromise, the utility assumes that a switch should occur unless

$$V(S, R) - \max_{x \neq R}(V(S, x) - C_{rx}) > \epsilon |V(S, R)| \qquad (25)$$

where ϵ is set to the default value of 10^{-5} (this value can be set by the user with the call `optset('osoptimum','tol',epsilon)`, where `epsilon` is any desired value). A useful check on the accuracy of this utility is to compare the output of the call

```
    regime=osoptimum(snodes,i,cv,model,fspace);
```

to the values of `x(:,i)` returned by `ossolve` (they should be identical).

Practicalities and Extensions

There are three choices an analyst must make in using `ossolve`: the family of approximating functions, the nodal values at which to solve the complementarity conditions and the initial coefficient values to use. As already mentioned, the use of smooth families of functions, like polynomials, is not recommended for switching models because of the discontinuities that occur in the second derivatives of the value function at the optimal switching points. Instead either cubic splines or piecewise linear functions with finite difference derivatives are better choices. These families can be defined using the CompEcon toolbox calls

```
      fspace=fundefn('spli',n,a,b);
```

or

```
      fspace=fundefn('lin',n,a,b);
```

In either case `n` is the number of basis functions needed to define the family and `a` and `b` are the lower and upper bounds of the approximation interval. When S is d-dimensional, `n`, `a` and `b` should all be $1 \times d$ vectors and tensor product basis functions will be formed from the basis functions for each dimension (see Miranda and Fackler (2002), chapter 6, for more details).

The syntax above defines cubic spline or piecewise linear functions with evenly spaced breakpoints on the interval $[a, b]$. The choice of the approximation interval is very important for the accuracy of the solution. If the interval is too wide, a large number of nodal points will be needed to obtain accurate solutions. If the interval is too small, endpoint problems may corrupt the solution. A general rule of thumb is that the interval should wide enough to include values to which the ergodic distribution of S (if it exists) assigns non-trivial probability. If no ergodic distribution exists (as with the widely used geometric Brownian motion), the interval should be large enough relative to the discount rate. Essentially this means that if the process starts near a switch point, the probability that it reaches the approximation boundary within short period of time is small. How short a period of time is appropriate in this calculation is determined by how fast the future is discounted. From a practical point of view, one can experiment with alternative values of a and b. These should be made extreme enough that the results of interest (generally the values of the switching points and the value function near these points) are relatively unaffected by the choice.

The CompEcon toolbox also allows one to specify spline or piecewise linear functions with unevenly spaced breakpoints. This may be useful if more accuracy is required. Putting more breakpoints in regions of non-smooth behavior, especially around the switch points, can result in much greater accuracy for the same order of approximation.

Some choice of nodal values must also be made. It is probably best to use the default values provided by the CompEcon Toolbox unless one has strong reasons for another choice. For splines and piecewise linear functions, the default nodal values are the breakpoints (with an extra point added near each end for cubic splines). Evaluation at the breakpoints ensures that the resulting basis matrices are well conditioned. The default nodal values used can be obtained prior to calling `ossolve` with the call[6]

```
snodes=funnode(fspace);
```

Starting values for the function coefficients can also be passed to the complementarity solver. If initial values are not passed, `ossolve` will compute default values by solving for the function coefficients that approximate a suboptimal value function with a decision rule that never switches the currently active regime. This may be a reasonable choice for starting values if nothing is known about the solution. If comparative static exercises are being performed by solving a model multiple times at a set of alternative model parameters, a good choice of

[6]For multidimensional state variables ($d > 1$), `snodes` is returned as a $1 \times d$ cell array of columns vectors. This can be transformed into a d-column matrix of points using `S=gridmanke(snodes)`.

starting values will generally be the values obtained from a previous call to `ossolve`.

Several extensions are of the basic model increase the flexibility with which it can be applied. First, suppose that, in addition to the decision maker choosing which regime is active, it is also possible that random exogenous shifts of regime occur. This possibility is described in greater detail in another paper, but suffice is to say here that a model of this type can be specified by defining two additional $m \times m$ matrices. The first of these, Λ, contains the Poisson jump intensities associated with an exogenous switch from regime i to regime j. The second matrix, Q, contains the costs imposed if such a jump occurs. Both matrices should have zeros on the diagonal.

To specify a model of this type, the `model` structure variable should contain two additional fields, `L` and `Q`, which contain the two matrices (if either `L` or `Q` is missing or empty, it is assumed to imply an $m \times m$ matrix of zeros). The only change in the algorithm is that the definitions of the M_i and q_i need to be appropriately modified.

Another extension that is straightforward is to allow the switching costs C, the Poisson intensities Λ and the costs due to exogenous switching Q to all be functions of S. Again, the solution approach is essentially unchanged except that the specific values of the M_i and q_i need to be adjusted appropriately. If the cost field is a string containing the word `'variable'` the solver calls the model function file with the flag `C`. The model function file should return an $n \times m$ matrix containing the cost of switching from regime x to the other regimes (this should contain a vector of zeros in column x). A similar syntax applies to Λ and Q, with the flag variable set to `L` and `Q`, respectively.

A third extension handles the situation in which the value function is known at some specified point or points. In such a case the complementary conditions (17) and (18) associated with the point (S, R) could be replaced by

$$\phi(S)\theta_R - V(S,R) \geq 0 \tag{26}$$

To implement this feature, the model variable should include a field named `values` containing a $d+2$ column matrix. Columns 1 through d are the values of S, column $d+1$ contains the regime number R and the last column contains the value of $V(S,R)$. For example suppose S is two-dimensional, with $V([S_1\ S_2], R)$. Setting

```
    model.values=[0 0 1 5;0 0 2 5]
```

will force $V([0\ 0], 1) = V([0\ 0], 2) = 5$. If all values in a specific dimension are involved, a `NaN` can be used in the associated column. For example, setting

```
    model.values=[0 nan 0 1;nan 0 1 2]
```

will set $V([0\ S_2], 1) = 0,\ \forall S_2$ and $V([S_1\ 0], 2) = 1,\ \forall S_1$. A value of `NaN` can also be used for the regime number, so

```
    model.values=[0 0 nan 5]
```

produces the same result as

```
    model.values=[0 0 1 5;0 0 2 5]
```

To force the numerical procedure to produce an approximate solution with known values, the optimality conditions (17) and (18) are replaced by condition (26) at the nodal points closest to each of the (S, R) values.

Finally, it may be desirable to allow for resetting of S when a regime switch occurs. For example, one of the states might measure the time spent in the current regime since the last regime change (so $dS = dt$). This state would be reset to 0 every time the regime changes. In general, if switching from regime i to regime j causes S to be reset to S_{ij}, condition (3) is modified to

$$V(S,i) \geq V(S_{ij}, j) - C_{ij} \tag{27}$$

and condition (18) to

$$\phi(S,i)\theta_i - \phi(S_{ij})\theta_j + C_{ij} \geq 0 \tag{28}$$

To implement this feature, the model variable should include a field named `reset` containing a $2 + d$ column matrix. The first d columns contain the target value of the state after resetting. Column $d+1$ contains the regime before the switch and columns $d+2$ is the regime after the switch. For example, if, on switching from regime $i = 2$ to regime $j = 3$, the state is reset to $S_{ij} = [0\ \ 1]$, use

```
    model.reset=[0 1 2 3]
```

If some of the variables are not reset upon switching, set the value of the state for these dimensions to `NaN`. For example, if only the first state variable is reset to 0 when switching from regime 1 to 2, use

```
    model.reset=[0 nan 1 2]
```

A Worked Example

This section demonstrates the application of the MATLAB procedures to the mine operation example discussed earlier (the demonstration files are included with the solver). The example first solves the model with a two dimensional state space (Q, P) and then solves the same model with the one-dimensional state space $y = QP$.

The first requirement is the model function file:

```
function out=minemodel2(flag,s,R,mu,sigma,rho,h,k)
  switch flag
    case 'f'
      out=(h*s(:,1).*s(:,2)-k).*(R==2);
    case 'mu'
      out=[-h*s(:,1).*(R==2) mu*s(:,2)];
    case 'sigma'
      out=[zeros(size(s,1),3) sigma*s(:,2)];
    case 'rho'
      out=rho;
  end
```

In addition to the required first three inputs, this function is defined in terms of five of the model parameters μ, σ, ρ, h and k (there are two additional model parameters C_{12} and C_{21}).

To solve the model, we also write a MATLAB script file that is called from the MATLAB command line. This file begins by defining the model parameters:

```
mu    = 0.01;
sigma = 0.02;
rho   = 0.04;
h     = 1;
k     = 2;
C12   = 5;
C21   = 2;
```

(the specific values are for demonstration purposes only). Next the model structure variable is defined:

```
clear model
model.func   = 'minemodel2';
model.params = {mu,sigma,rho,h,k};
model.cost   = [0 C12;C21 0];
```

Then the family of approximating functions is defined:

```
fspace=fundefn('lin',[51 51],[0 0],[100 10]);
```

Here a piecewise linear function with 51 evenly spaced breakpoints for Q on [0,100] and 51 evenly spaced breakpoints for P on [0,10] are used (this family of approximating functions uses finite difference derivatives). The solver is now called:

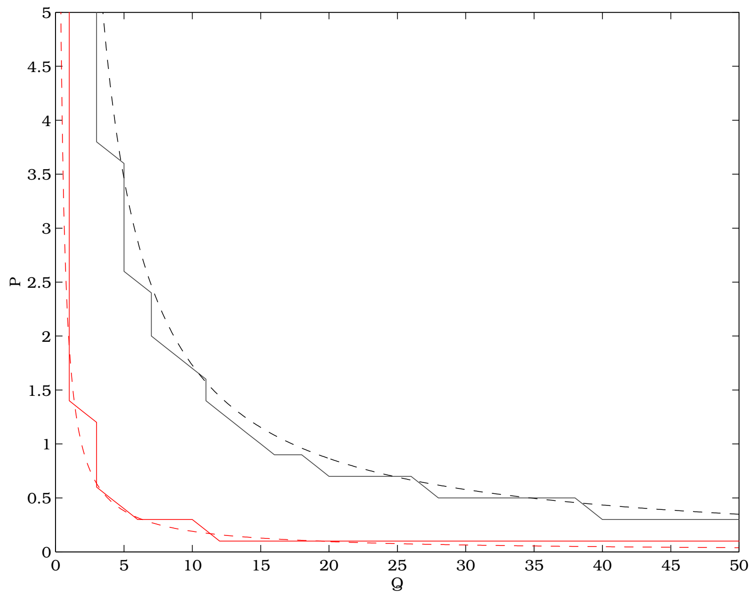
```
[cv,snodes,x]=ossolve(model,fspace);
```

Using the output, a plot of the optimal switch boundaries can be computed, as is shown in the solid lines in Figure 1. The lower line represents the points for which it is optimal to switch from active to inactive, the upper line represents the points for which it is optimal to switch from inactive to active. The jaggedness in an inevitable consequence of the discreteness of the nodal values. Although it might be useful to smooth these curves, no attempt has been made to do so here.

The model can also be solved using the one dimensional reformulation. In this case the model function file would be written

```
function out=minemodel1(flag,s,R,mu,sigma,rho,h,k)
  switch flag
    case 'f'
      out=(h*s-k).*(R==2);
    case 'mu'
      out=(mu-h*(R==2))*s;
    case 'sigma'
      out=sigma*s;
    case 'rho'
      out=rho;
  end
```

Figure 1: Optimal Switch Boundaries for the Mine Example



The model variable `func` field would be changed to the name of this procedure

```
model.func = 'minemodel1';
```

Also the `fspace` variable would be altered

```
fspace=fundefn('lin',501,0,50);
```

This defines a family of piecewise linear functions with 501 breakpoints on [0 50]. When calling the solver, it will be useful now to obtain the fourth output `xindex`:

```
[cv,snodes,x,xindex]=ossolve(model,fspace);
```

Elements (1,4) and (2,1) of `xindex` contain the approximate locations of the switch points for the inactive and active regimes, respectively. For the parameter values given above, these points are computed to be 17.3 and 1.9. The optimal switchpoints are therefore approximately points satisfying $QP = 17.3$ and $QP = 1.9$, which are shown in the dashed lines in Figure 1.

The one-dimensional mine example has an almost explicit solution, which can be used to obtain highly accurate optimal switching boundaries and value functions (see, e.g., Dixit and Pindyck for discussion of this approach). To four decimal places, the switching boundaries are 17.2522 and 1.9233. For practical purposes, this is indistinguishable from the one-dimensional numerical solution. Furthermore, the value function approximation was accurate to approximately four significant digits.

Summary

This paper describes a general numerical approach to solving optimal switching problems and documents a MATLAB based implementation of the approach. The basic framework consists of a model for a stochastic process S that characterized by its drift and diffusion functions μ and σ, by a stream of rewards described by the function f, by a discount rate ρ (possibly state contingent) and by a switching cost matrix C. The solution approach requires that these parameters be specified along with a family of approximating functions. The solution algorithm can then set up and solve a set of complementarity conditions that are satisfied at the problem solution.

The solution approach described and implemented here has a number of important advantages over previously described solution approaches. First, it is generic and hence the code required to solve specific models is mainly limited to specifying the functions and parameter values that define the model, along with code to call the solver and interpret the solver output. Second, it can solve models with general multidimensional diffusion processes. This makes it easy to solve models without the restriction to one-dimensional geometric Brownian motion found in much of the existing literature. Third, unlike the generic one-dimensional solver described in Miranda and Fackler (chap. 11), the solution approach used here does not require that the analyst guess at the qualitative nature of the optimal solution. In particular, it is not necessary to know to which regime it is optimal to switch at the boundaries of the no-switch regions.

References

Brekke, Kjell Arne and Bernt Oksendal. “Optimal Switching in an Economic Activity Under Uncertainty.” *SIAM Journal on Control and Optimization* 32(1994):1021–1036.

Brennan, Michael J. and Eduardo S. Schwartz. “Evaluating Natural Resource Investments.” *Journal of Business* 58(1985):135–157.

Brennan, Micheal J. and Lenos Trigeorgis, editors. *Project Flexability, Agency and Competition*. Oxford: Oxford University Press, 2000.

Cottle, R.W. and G.B. Dantzig. “A Generalization of the Linear Complementarity Problem.” *Journal of Combinatorial Theory* 8(1970):79–90.

Dixit, Avinash K. and Robert S. Pindyck. *Investment Under Uncertainty*. Princeton, NJ: Princeton University Press, 1994.

Dumas, Bernard. “Super Contact and Related Optimality Conditions.” *Journal of Economic Dynamics and Control* 15(1991):675–685.

Gowda, M. Seetharama and Roman Sznajder. “The Generalized Order Linear Complementarity Problem.” *SIAM Journal of Matrix Analysis and Applications* 15(1994):779–795.

Judd, Kenneth L. *Numerical Methods in Economics*. Cambridge, MA: MIT Press, 1998.

Lemke, C.E. “Bimatrix Equilibrium Points and Mathematical Programming.” *Management Science* 11(1965):681–689.

McDonald, Robert L. and Daniel R. Siegel. “Investment and the Valuation of Firms When There is an Option to Shut Down.” *International Economic Review* 26(2)(1985):331–349.

McDonald, Robert L. and Daniel R. Siegel. “The Value of Waiting to Invest.” *Quarterly Journal of Economics* 101(1986):707–727.

Miranda, Mario J. and Paul L. Fackler. *Applied Computational Economics and Finance*. Cambridge MA: MIT Press, 2002.

Qi, Hou-Duo and Li-Zhi Liao. “A Smoothing Newton Method for Extended Vertical Linear Complementarity Problems.” *Journal of Matrix Analysis and Applications* 21(1999):45–66.

Schwartz, Eduardo S. and Lenos Trigeorgis, editors. *Real Options and Investment Under Uncertainty*. Cambridge, MA: MIT Press, 2001.

Sun, Min. “Monotonicity of Mangasarian’s Iterative Algoithm for Generalized Linear Complementarity Problems.” *Journal of Mathematical Analysis and Applications* 144(1989):474–485.