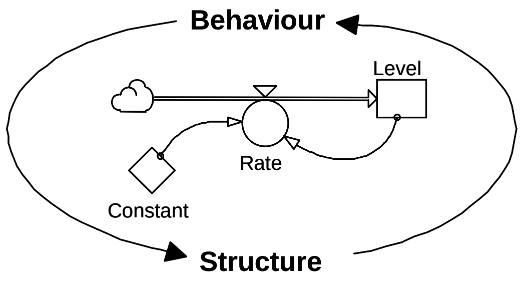
WORKING PAPERS IN SYSTEM DYNAMICS

WPSD 2/05 **ISSN 1503-4860**

SOPS – A Tool to Find Optimal Policies in Stochastic Dynamic Systems

by

Arne Krakenes
Erling Moxnes



The System Dynamics Group
Department of Geography
University of Bergen
Fosswinckelsgt.6, N-5007 Bergen, Norway

SOPS – A Tool to Find Optimal Policies in Stochastic Dynamic Systems

Arne Krakenes
System Dynamics Group, University of Bergen and Powersim Software AS
Powersim, Sandbrugaten 5-7, PO Box 3961,
Dreggen N-5835 Bergen, Norway
Tlf.: +47 55 60 65 00
arne.kraakenes@powersim.no

Erling Moxnes
System Dynamics Group, University of Bergen, Norway
Department of Geography, Fosswinckelsgt.6
N-5007 Bergen, Norway
Tlf.: +47 55 58 41 19
Erling.Moxnes@ifi.uib.no

Abstract

The task of finding optimal policies in stochastic dynamic systems is challenging. The theory of stochastic dynamic programming (SDP) is quite complex and the available software packages are not intended for non-specialists. Furthermore, SDP is traditionally limited to quite small and well defined problems. Stochastic optimisation in policy space (SOPS) seems to be an attractive alternative, particularly for people with a background in simulation of dynamic systems. However, to date no user friendly software has been available for this method. In this paper we present and demonstrate a new program package for this task. The resulting software allows the user to formulate the model in a well-known simulation program, Powersim Studio 2005. The model is automatically transferred to a standalone program. The SOPS program allows the user to reset model parameters, to specify search criteria, and to study the results of repeated searches for optimal policies.

Key words: stochastic dynamic optimization, user friendly, Monte Carlo, search, optimization in policy space

1. INTRODUCTION

When making decisions in stochastic dynamic systems we face two classes of complexity, both leading to problems in understanding the implications of our decisions. First, there is complexity caused by the dynamic nature of systems. Several experimental studies have shown that we have problems making the right decisions in dynamic systems (e.g. Sterman (1989) and Moxnes (2004)). Secondly, similar studies have shown that we also have problems making good judgements under uncertainty (e.g. Tversky and Kahneman (1974)) and Kahneman and Tversky (1979)). In spite of these difficulties we have to manage stochastic dynamic systems.

As a first attempt, strategies could be tested by simulation. This method allows for creative solutions and could bring a system a long way from mismanagement towards proper management. It is however difficult to take account of uncertainty with this method. For this purpose, Monte Carlo simulations help test strategies over wide ranges of possible futures. To make the search for the best possible strategies more efficient, the search process can be automated. The program that we describe in this paper does that. The method is called stochastic optimisation in policy space (SOPS) and that is also the name we have given to the program package that is developed for this purpose.

The traditional method for this optimisation problem is called stochastic dynamic programming (SDP). (Analytical methods are also available to solve highly simplified problems.) SDP has proven very useful for many problems and could in many cases be a useful supplement to SOPS. However, there are also some drawbacks with SDP that make SOPS an interesting alternative, particularly for infinite horizon problems. According to Lubow (1995):

> "A significant drawback to the dynamic programming technique is the high computational requirements for problems with more than a few state or decision variables ("the curse of dimensionality," Bellman 1957). However, the advent of inexpensive, high performance personal computers and workstations has significantly reduced this problem. Undoubtedly, the highly theoretical and mathematical nature of some literature on dynamic programming has also impeded application of this technique (see the discussion of this problem in Nemhauser [1966:245]). However, the absence of adequate software may pose the most significant obstacle facing potential users of the dynamic programming technique. Morin (1979) identified 10 "fairly general" dynamic programming codes; however, these are now >15 years old, mainframe based, and designed to solve deterministic dynamic programming problems. Labadie (1990) developed CSUDP, a generalized software tool for microcomputers that incorporates several sophisticated techniques for reducing computation time for large deterministic problems. CSUDP can also solve small (2-state variable) stochastic problems; however, this package can not effectively solve larger stochastic problems."

Lubow goes on to describe his own user friendly package (called SDP), however noting that:

> "it can not replace user's understanding of the conceptual basis of the technique ---. SDP should not be viewed as a means of providing novices with an easy recipe for solving complex dynamic optimisation problems. Rather, it is intended to assist investigators familiar with dynamic programming ---"

SOPS may help remove or reduce these difficulties. Rather than solving the problem backwards in time, which is the technique in SDP, SOPS uses repeated simulations forward in time with different sequences of random variables (Monte Carlo). The average criterion obtained represents the expected criterion value. This procedure is repeated while systematically changing the proposed policy until the expected criterion value is maximised. This procedure is similar to what is going on when searching for good policies by trial-and-error.

While optimisation in policy space has been described and discussed by several authors, see references in Moxnes (2003), we have found no user-friendly and efficient software to perform the automated policy search. Therefore we have developed the SOPS program. The user formulates the underlying dynamic model in a well-known software package, Powersim Studio 2005. By one command, the Powersim Studio model is translated into C+ code, this code is compiled and made ready for the SOPS program, and then the SOPS program opens and is ready to perform searches for optimal policies. In order to represent policies in a generic way to enable policy optimisation, we also add some functionality to Powersim Studio itself.

Stochastic optimisation can be useful for many purposes. The obvious purpose is to find optimal policies for decision makers. However, in this case the practical usefulness depends on the client's awareness of the problem and willingness to rely on more or less black box numerical results. Other purposes may be less obvious, however, may prove to be of great practical value for analysts. Strategies found by intuition and repeated simulations may be checked and possibly altered before they are used to update clients' mental models and heuristics. Moxnes (2003) for instance show how strategies for the total allowable catch of a fish species should be altered when measurement error is introduced. Stochastic optimisation may be used for policy sensitivity analysis, where the sensitivity of the optimal policy to uncertain model assumptions is tested, see Moxnes (2005). When performing laboratory experiments, optimal benchmark strategies and results can be found and used to judge observed subject behaviour.

In the next section we give a short presentation of the method. Then an example is used to demonstrate the method and the program.

2. STOCHASTIC OPTIMISATION IN POLICY SPACE

To explain SOPS we start by explaining optimisation in policy space (OPS). We use a simple fishery model as an example. The challenge is to find a harvesting policy that yields the highest possible value of accumulated profits at the end of the simulation. This is illustrated in Figure 1.

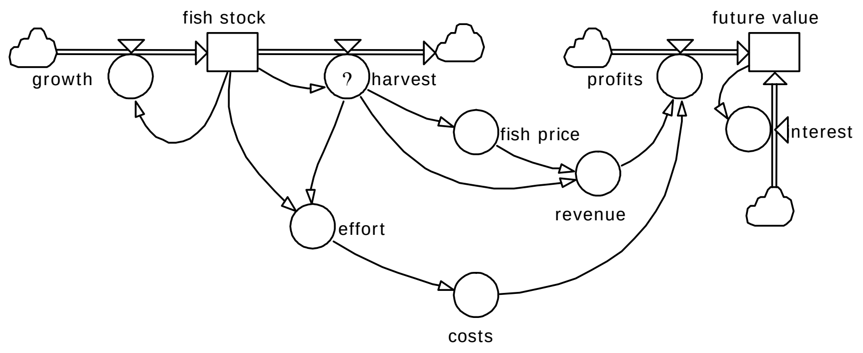


Figure 1: The fishery model

Accumulated profits are represented by their future value. Maximising the future value is the same as maximising the present value. While the latter criterion is usually used in dynamic programming, the future value seems to be the most intuitive choice when using simulation. The interest rate implies that the first years of the simulation weight more than the later years. Low weights on future years mean that an infinite horizon can be approximated by a limited number of time periods.

To simplify let us assume a harvesting policy

$$H(x)=\theta x=f(x,\theta)$$

where the harvest H is proportional to the fish stock x, where θ is a *policy parameter*, and where we denote $f(x,\theta)$ the *policy function*. Furthermore we denote the future value of profits W. For each choice of policy parameter θ there will be a corresponding criterion value W. Hence the dynamic optimisation problem is reduced to finding the policy parameter θ^* that maximises W. Figure 2 illustrates. The optimal policy can be found by repeated simulations in a trial-and-error fashion, or one can automate the process by using a numerical search algorithm.

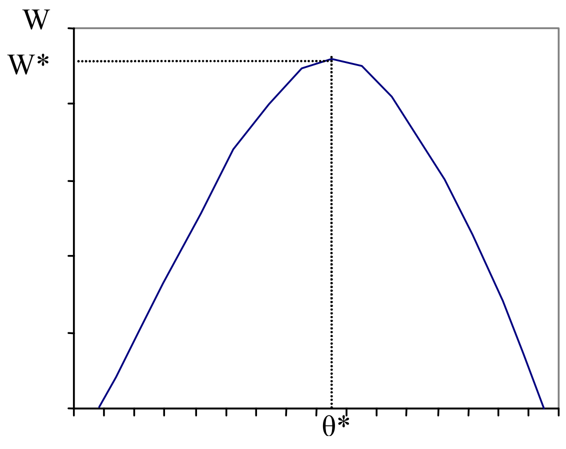


Figure 2: Policy - criterion graph

Policies are usually more complex than the one-parameter policy above. At the outset the structure of the policy function is not known. In spite of this, optimisation in policy space requires that the modeller specifies a policy function structure before the search for policy parameters starts. If one chooses a “fixed” policy function, the search can be handled by optimisation tools available in both Powersim Studio and VenSim. However, these tools may not provide the functional flexibility needed to identify true optima. Since we do not know the functional form a priori, the ideal policy function is fully flexible.

A grid function with a sufficient number of grid points provides flexibility. For the one dimensional case, where there is only one input variable to the policy, we can use the inter- and extrapolating graph function available in most simulation tools. In Powersim Studio, this function uses the following syntax:

$$H(x)=GRAPHLINAS(x,\varphi,\delta,\theta)$$

In addition to the symbols that we have already defined, φ defines the position of the first grid point and δ denotes the distance between the grid points, see Figure 3.

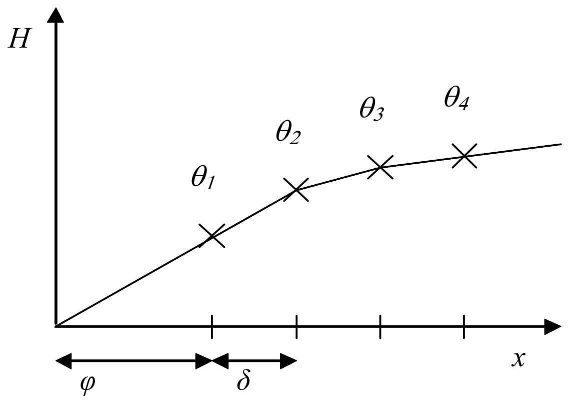


Figure 3: Graph Function in the policy space

The policy function in Figure 3 is only appropriate when the harvest is a function of only one input variable. In principle, optimal policies are functions of all stocks (state variables) in a model. Therefore it is important to be able to test out policy functions that take more than one input variable. For this purpose we have added a new function to the Powersim Studio library of functions

$$H(x) = POLICYGRID(\theta, \lambda, \varphi, \delta, x)$$

Now, x, φ, and δ are all vectors. The θ-vector represents all the grid points in the policy surface in as many dimensions as there are input variables x. In addition, the vector λ can be used to denote upper and lower limits for $H(x)$. We illustrate with a two dimensional example in Figure 4. We denote the input variables x and y and we choose four grid points in the x-direction and three in the y-direction. Intervals between grid points are all 1.0. The table shows the θ -values and the graph shows the policy surface. The *POLICYGRID* -function interpolates between the grid points, and extrapolates beyond the outer grid points.

y\x	1	2	3	4
1	1	2	2	3
2	2	5	3	3
3	4	6	8	9

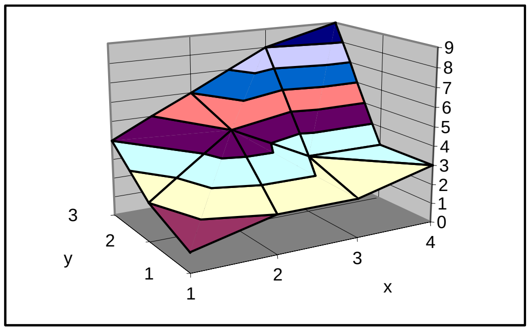


Figure 4: Table with θ-values and graph showing interpolated policy surface.

Thus far we have dealt with deterministic dynamic problems. The introduction of stochasticity requires one more step. Figure 5 illustrates how fish growth is influence by random environmental variations.

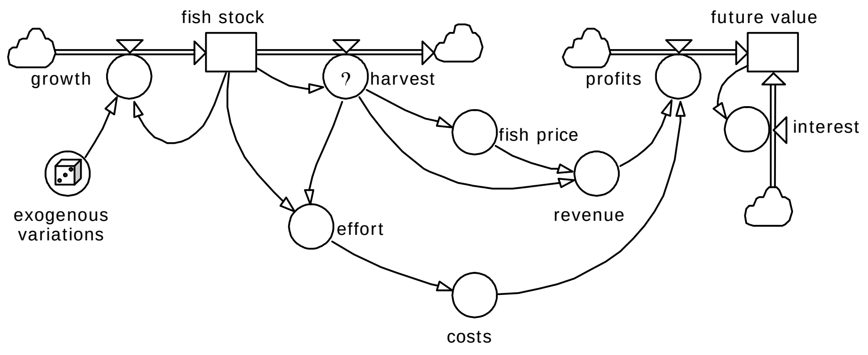


Figure 5: Fishery model with stochastic growth

The stochastic variable "exogenous variations" gives different growth rates each year. This causes the criterion to be unpredictable. If one uses the criterion value *W* after one simulation to search for the optimal policy parameter vector *θ*, one can no longer know whether an improvement in *W* is caused by a change in *θ* or by the outcomes of the random variables. To avoid this problem one can use the same seed for the random variables each time one simulates. Then *W* will no longer change from one simulation to the next unless *θ* is changed. However, in this case the optimal policy will be adjusted to future outcomes of the random variables, outcomes than one does not have information about before they occur. To avoid this problem we use Monte Carlo simulations, such that the policy is tested against a series of possible future outcomes of the random variables. We use the letter *J* to denote the new criterion being the average of the criterion values $W(\theta)_m$ from the *M* Monte Carlo simulations:

$$J(\theta) = \frac{1}{M}\sum_{m=1}^{M} W(\theta)_m$$

Each time *J* is evaluated, the same seed is used for the random generator. Thus *J* is predictable and only a function of *θ*. Hence, the entire stochastic dynamic optimisation problem is transformed into a deterministic nonlinear search problem similar to what we illustrated in Figure 2.

Further details about the method can be found in Moxnes (2003) and Moxnes (2005). Here we just summarize some key points. First, to rule out that the search algorithm ends up in a local optimum, repeated searches should be carried out with different starting values for *θ*. Second, in principle all stocks (state variables) could be important as inputs to the policy function. In case of measurement error, historical measurements may also be important.

Third, to ensure that the policy function is not restricting the solution, rapidly increasing numbers of grid points are needed when the number of input variables increases. Thus, there is also a 'curse of dimensionality' when using grid policies in SOPS. To be able to search for policies in the high dimensionality case, we have added a policy option in Powersim Studio called *SOPSCUSTOMPOLICY.* This function allows SOPS to search for policy parameters in any analytical function. In this case SOPS finds simplified policies, and not ideal policies, for highly complex dynamic problems. This is an interesting alternative to model simplification, which it typically resorted to by those who use stochastic dynamic programming and prioritise finding truly optimal policies. Particularly if simplified policies represent the only practical option for policy makers, the SOPS program comes in handy since it helps finding policy parameters that maximise the criterion for different suggestions about practical policies.

It is possible to specify several policies in one and the same model (both grid and custom policies). Then SOPS maximise the criterion for all policies jointly. In complex cases it may be practical to start with custom policies to establish feasible initial θ-values for grid policies to be used in later stages of the analysis where the policy is refined.

3. PRACTICAL USE OF SOPS

Then we are ready to demonstrate the SOPS program using the earlier fishery example. Figure 6 shows the Powersim Studio 2005 representation of a discrete-time version of the model.[1]

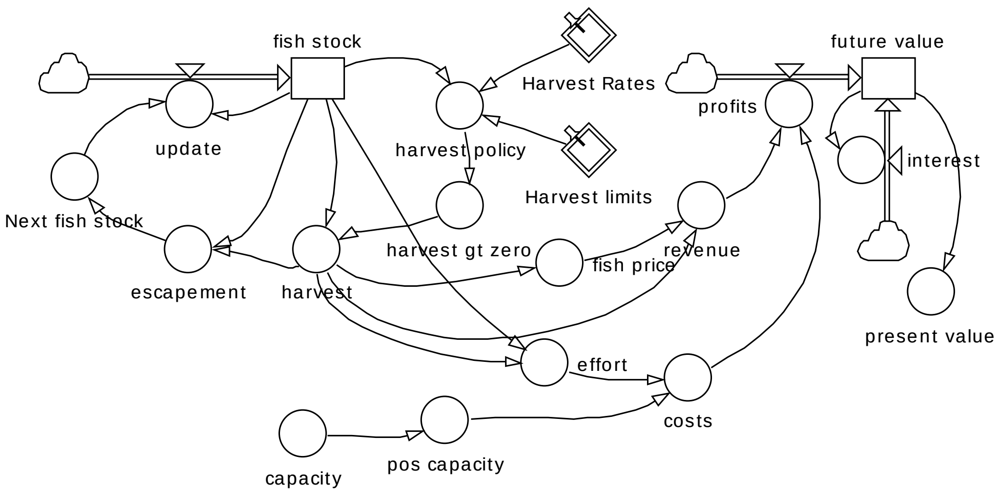
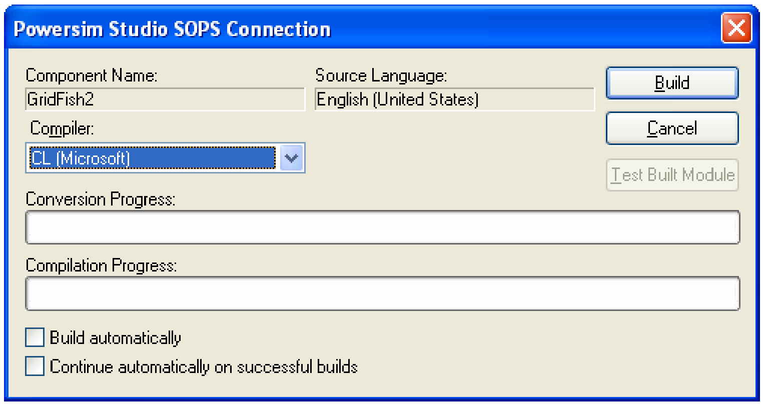


Figure 6: Revised Fishery Model

After the model has been built in Powersim Studio, we activate the SOPS tool from within Studio, and enter the Studio-SOPS connection dialog where we build the compiled simulation model:

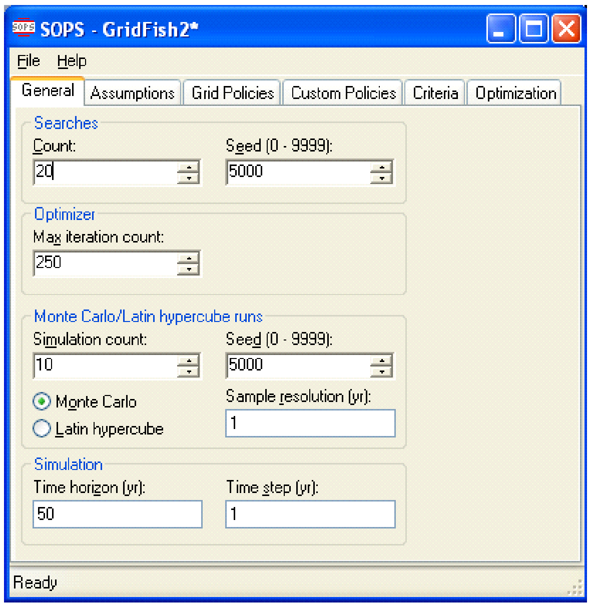
1 *Escapement* is the difference between the beginning of year (pre catch) fish stock and (all at beginning of year) harvest. *Next fish stock* is the fish stock after both (beginning of year) harvest and (throughout the year) growth. In this representation it is easy to prevent the fish stock from going negative, or below a given value, by using a max-function in the equation for *Next fish stock*.


Powersim Studio SOPS Connection

Component Name: GridFish2
Source Language: English (United States)
Compiler: CL (Microsoft)
Conversion Progress:
Compilation Progress:
☐ Build automatically
☐ Continue automatically on successful builds

Figure 7: Converting Fishery Model

When the build is finished, it is tested automatically to make sure the compiled simulation model gives the same results as the Studio simulation model. Differences could appear because there are a few advanced features in Powersim Studio that are not implemented in the SOPS program. The program identifies the causes. Furthermore, minor numerical differences may occur because Powersim Studio has routines to avoid round off errors. These can be ignored and we may proceed to the SOPS application:


SOPS - GridFish2*

Searches
Count: 20
Seed (0 - 9999): 5000

Optimizer
Max iteration count: 250

Monte Carlo/Latin hypercube runs
Simulation count: 10
Seed (0 - 9999): 5000
[✓ Monte Carlo] Latin hypercube
Sample resolution (yr): 1

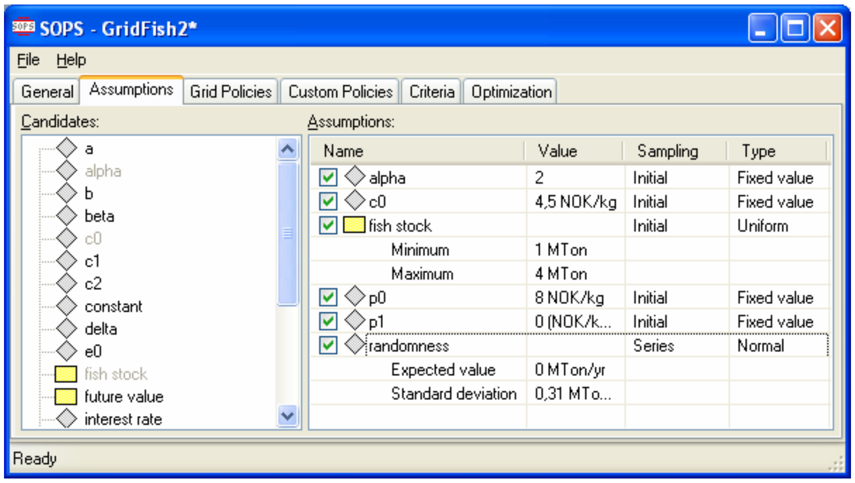
Simulation
Time horizon (yr): 50
Time step (yr): 1

Ready

Figure 8: General Settings

Figure 8 shows the opening page for SOPS with tags for the other pages. Having in mind the local and global optima mentioned in section 2, we decide to do many rough searches (20) with a quite broad set of initial policies. We use few Monte Carlo runs (10) to speed up the searches at this stage.

The "Assumptions" page in Figure 9 is used to define the model assumptions. One can set the values of constants and initial values. The assumptions may be deterministic (fixed value) or drawn from distributions. The stochastic values may be drawn initially or over time (Series). Here we define a normal distribution for series of random environmental impacts on growth and we specify a uniform distribution for the initial fish stock. We also set fixed parameters for the cost and price functions (c_0, p_0, and p_1).



SOPS - GridFish2*

File Help

General | Assumptions | Grid Policies | Custom Policies | Criteria | Optimization

Candidates: a, alpha, b, beta, c0, c1, c2, constant, delta, e0, fish stock, future value, interest rate

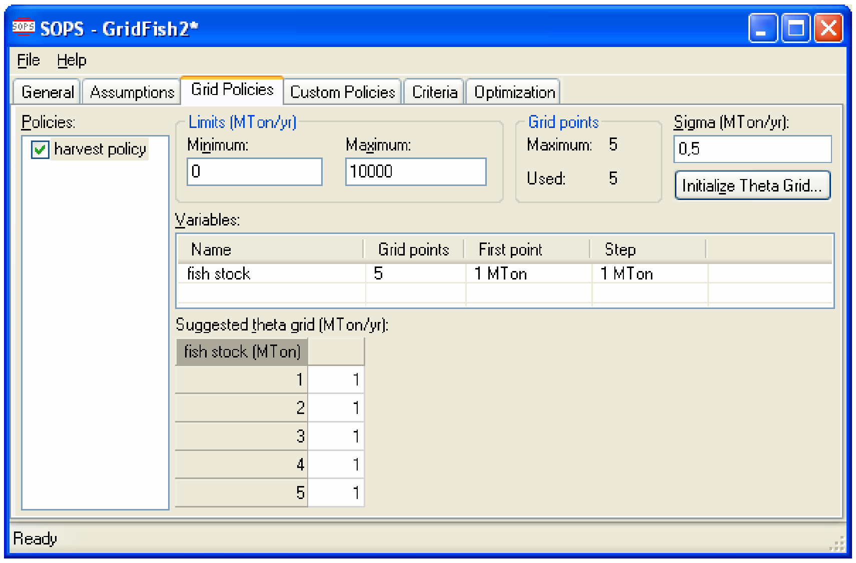
Assumptions:

Name	Value	Sampling	Type
✓ alpha	2	Initial	Fixed value
✓ c0	4,5 NOK/kg	Initial	Fixed value
✓ fish stock		Initial	Uniform
Minimum	1 MTon		
Maximum	4 MTon		
✓ p0	8 NOK/kg	Initial	Fixed value
✓ p1	0 (NOK/k...	Initial	Fixed value
✓ randomness		Series	Normal
Expected value	0 MTon/yr		
Standard deviation	0,31 MTo...		

Ready

Figure 9: Assumptions

The fishery model we use in this test contains one grid policy, the harvest policy. Entering the “Grid Policies” page in Figure 10, we see that the tool displays this policy (the program recognises the grid policy function in Powersim Studio 2005) and chooses this option. We enter the initial θ-values, either manually, by paste or automatically (the Initialize Theta Grid function produces a plane). The number of grid points, position of the first grid point and the step can be changed. Finally, the Sigma value determines the standard deviation for the distribution of initial θ-values.



SOPS - GridFish2*

File Help

General | Assumptions | Grid Policies | Custom Policies | Criteria | Optimization

Policies: ✓ harvest policy

Limits (MTon/yr) Minimum: 0 Maximum: 10000

Grid points Maximum: 5 Used: 5

Sigma (MTon/yr): 0,5

Initialize Theta Grid...

Variables:

Name	Grid points	First point	Step
fish stock	5	1 MTon	1 MTon

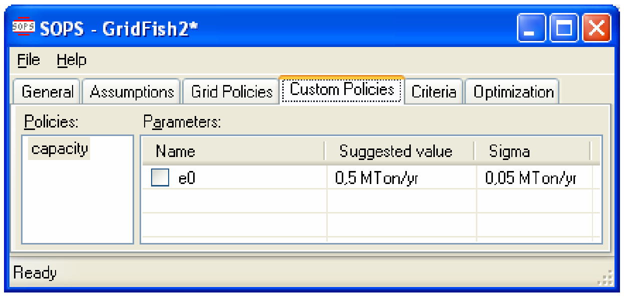
Suggested theta grid (MTon/yr):

fish stock (MTon)	
1	1
2	1
3	1
4	1
5	1

Ready

Figure 10: Grid Policies

Then we move on to the “Custom Policies” page in Figure 11. Although we have formulated a custom policy for capacity in Powersim Studio (the constant e_0), we do not want to optimise the value of e_0 now. Therefore we uncheck this policy, and fix the value of e_0 at 0.5 MTon/yr:


SOPS - GridFish2*
File Help
General Assumptions Grid Policies Custom Policies Criteria Optimization
Ready

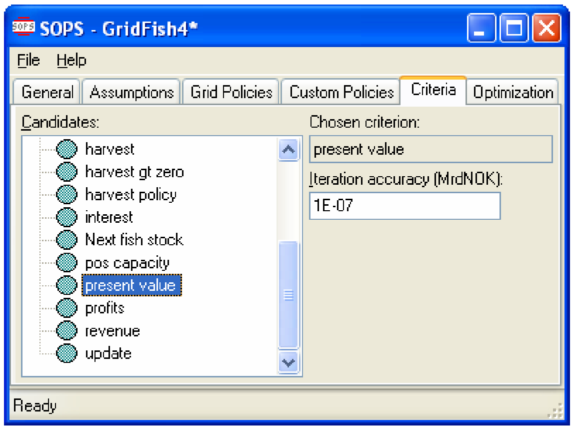
Policies: capacity

Parameters:

Name	Suggested value	Sigma
☐ e0	0,5 MTon/yr	0,05 MTon/yr

Figure 11: Custom Policies

The last specification page is the "Criterion" page, Figure 12, where we choose the criterion and the accuracy of the optimisation. All model variables present themselves as alternatives and we choose "Present Value" as our criterion. We do not change the suggested accuracy.


SOPS - GridFish4*
File Help
General Assumptions Grid Policies Custom Policies Criteria Optimization
Ready

Candidates:
- harvest
- harvest gt zero
- harvest policy
- interest
- Next fish stock
- pos capacity
- present value
- profits
- revenue
- update

Chosen criterion: present value

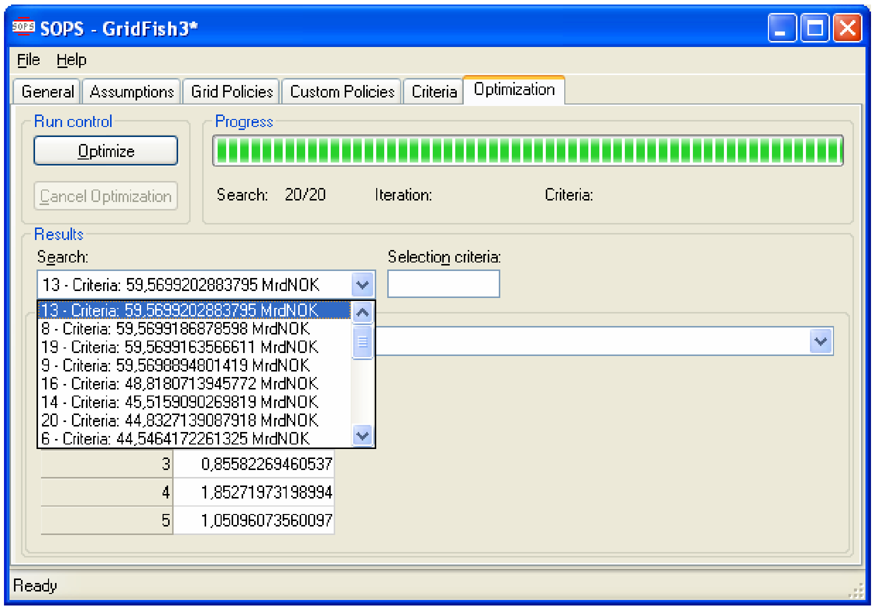
Iteration accuracy (MrdNOK): 1E-07

Figure 12: Selecting Criterion

Now we are ready to run the optimisation, so we enter the last page of the SOPS application, the "Optimisation" page in Figure 13. We simply click on optimise. The optimisation finishes rapidly and we open the list of searches.

We see that four of the searches give much better criterion values than the rest and we select these searches by entering 59 in the field called "Selection criterion". Now the program only displays the four searches with the highest criterion values. It also displays the average of those four searches and the standard deviation of the results. These two options can be selected by scrolling down the list in the "Search" field.

Once a particular search result, the average result, or the standard deviation is selected, the corresponding policy parameter values are shown in the table at the bottom of the page. Figure 14 shows averages. The two first policy parameters (θ_1 and θ_2) are negative and are overruled by a lower limit (λ) of zero for harvests. At higher stock levels from 2 to 4 the harvest increases nearly linearly. The program has identified the well known "target escapement" policy for this problem. The result would be even closer to linear if the number of Monte Carlo simulations had been increased (now the policy is slightly adapted to the limited number of future scenarios), and if the time horizon had been extended beyond 50 years.



Search: 20/20 Iteration: Criteria:

Search: 13 - Criteria: 59,5699202883795 MrdNOK

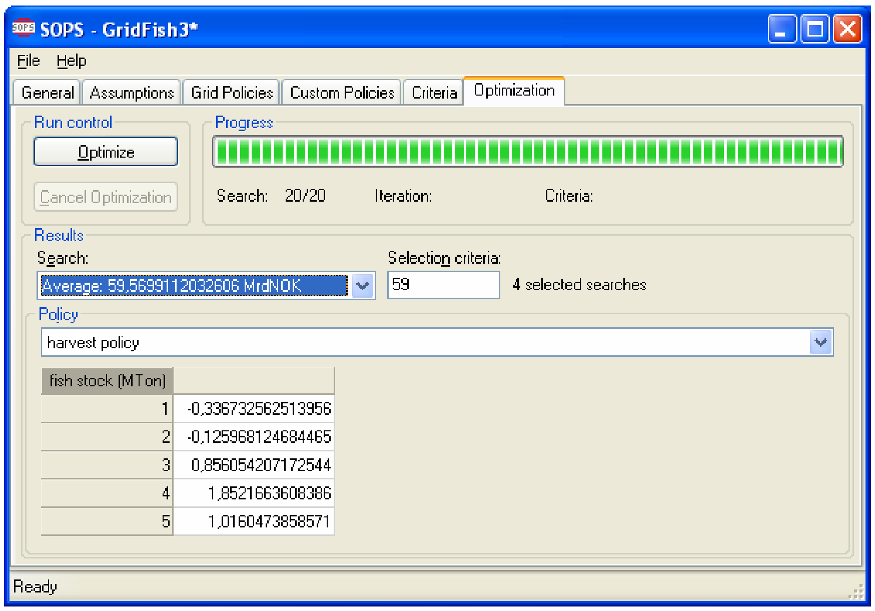
Selection criteria:

```
13 - Criteria: 59,5699202883795 MrdNOK
8 - Criteria: 59,5699186878598 MrdNOK
19 - Criteria: 59,5699163566611 MrdNOK
9 - Criteria: 59,5698894801419 MrdNOK
16 - Criteria: 48,8180713945772 MrdNOK
14 - Criteria: 45,5159090269819 MrdNOK
20 - Criteria: 44,8327139087918 MrdNOK
6 - Criteria: 44,5464172261325 MrdNOK
```

3	0,85582269460537
4	1,85271973198994
5	1,05096073560097

Ready

Figure 13: Optimisation



Search: 20/20 Iteration: Criteria:

Search: Average: 59,5699112032606 MrdNOK

Selection criteria: 59 4 selected searches

Policy: harvest policy

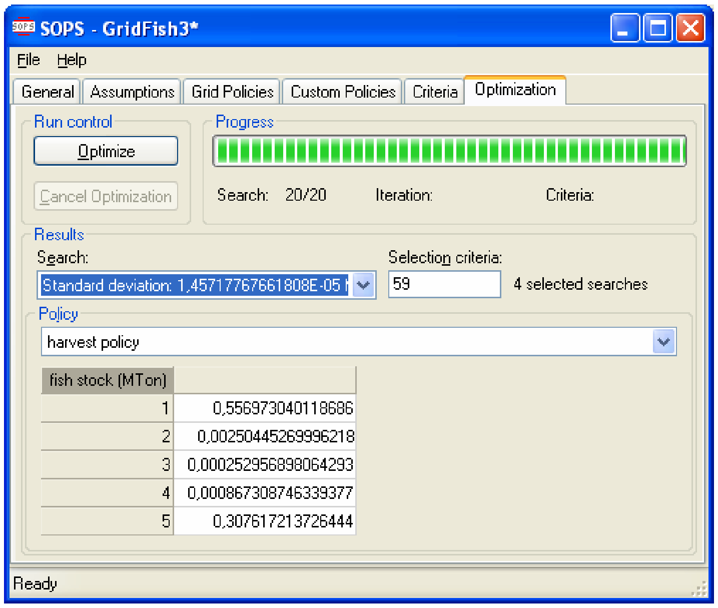
fish stock (MTon)	
1	-0,336732562513956
2	-0,125968124684465
3	0,856054207172544
4	1,8521663608386
5	1,0160473858571

Ready

Figure 14: Average Optimisation Results

We select the “Standard Deviation” entry of the “Search” field in Figure 15, and see that the standard deviations of the first and last value in the grid are quite large.

The higher the standard deviation is over the searches, the more sceptical one should be of the individual search results and of the average results. For fish stocks in the range from 3 to 4 the accuracy is great. In the range from 1 to 2, the policy is negative and is overruled by the lower limit of zero. Hence, θ_1 does not influence the criterion and its value is random. Parameter θ_2 has only a minor effect on the policy, still the accuracy is fully acceptable. Perhaps more surprising, the policy parameter at a fish stock of 5 is inaccurate. Here the reason is that with this policy, the fish stock only rarely exceeds 4. Hence the policy parameter at a fish stock of 5 is of only minor importance for the criterion and therefore it is not determined with any precision.



Search: 20/20 Iteration: Criteria:

Search: Standard deviation: 1,45717767661808E-05

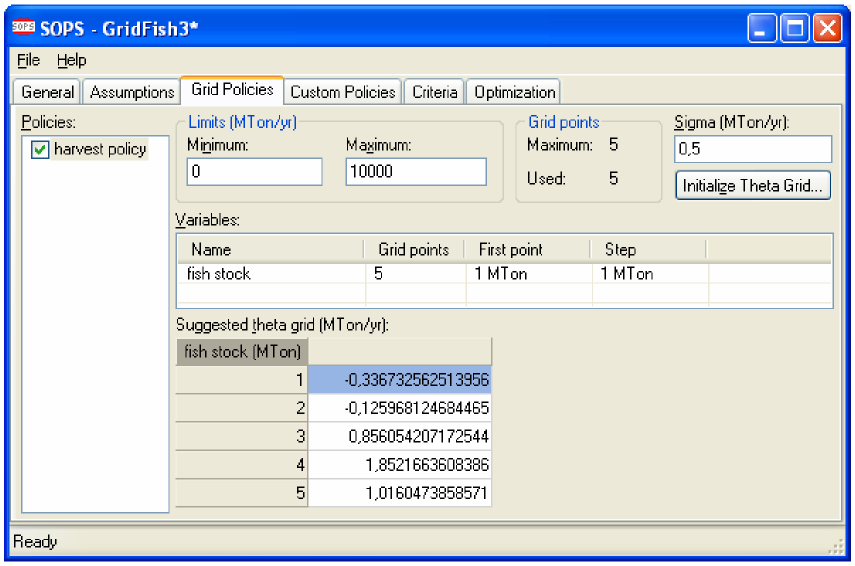
Selection criteria: 59 4 selected searches

harvest policy

fish stock (MTon)	
1	0,556973040118686
2	0,00250445269996218
3	0,000252956898064293
4	0,000867308746339377
5	0,307617213726444

Figure 15: Standard Deviation

We reselect the average, and mark all θ-values in the theta grid table, using the mouse, and copy them (Ctrl+C). Then we return to the “Grid Policies” page, and paste the values into the “Suggested Theta Grid” table, Figure 16:


SOPS - GridFish3*
File Help
General Assumptions Grid Policies Custom Policies Criteria Optimization
Initialize Theta Grid...
Ready

Policies: ✓ harvest policy

Limits (MTon/yr): Minimum: 0 Maximum: 10000

Grid points: Maximum: 5 Used: 5

Sigma (MTon/yr): 0,5

Variables:

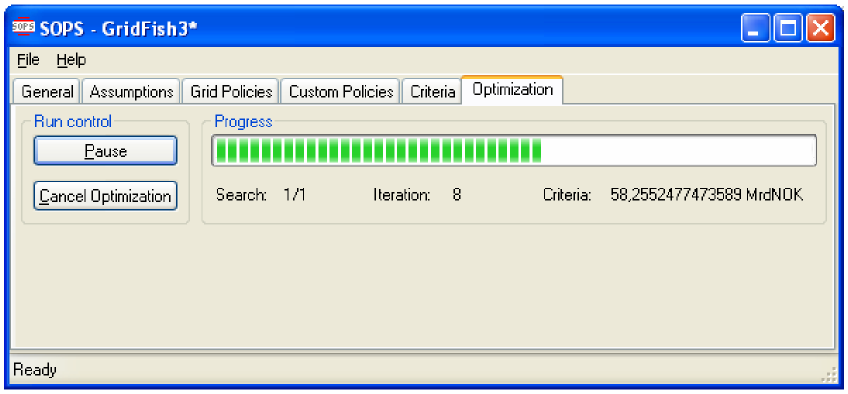
Name	Grid points	First point	Step
fish stock	5	1 MTon	1 MTon

Suggested theta grid (MTon/yr):

fish stock (MTon)	
1	-0,336732562513956
2	-0,125968124684465
3	0,856054207172544
4	1,8521663608386
5	1,0160473858571

Figure 16: Grid Policies Revisited

We also return to the “General” page and change the number of Monte Carlo simulations to 10000 and the search count to 1. When doing only one search, the suggested theta grid will be used “as is” (no random deviation from the suggested). Hence our new search will be initialized with the exact policy found by the rough initial optimisation. Using the new settings, we start a new optimisation. This time the optimisation is much slower, so we even manage to get a screenshot before the first search is finished, Figure 17:

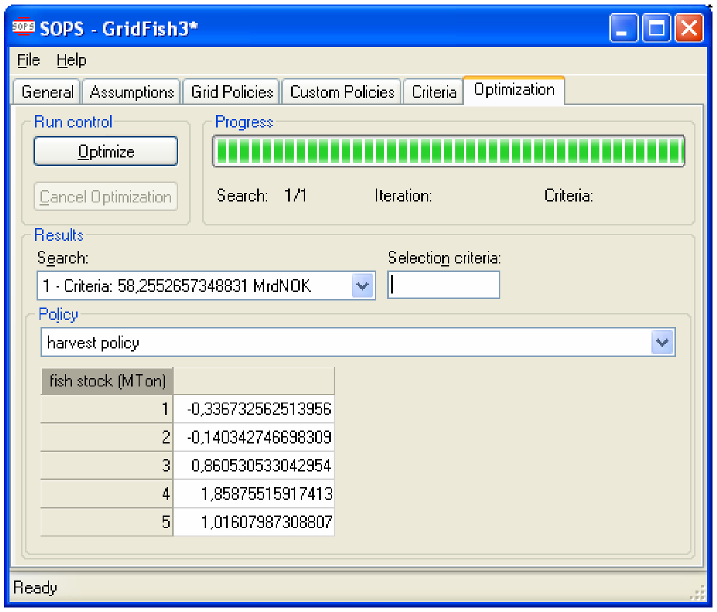

SOPS - GridFish3*
File Help
General Assumptions Grid Policies Custom Policies Criteria Optimization
Run control
Pause
Cancel Optimization
Progress

Search: 1/1 Iteration: 8 Criteria: 58,2552477473589 MrdNOK

Ready

Figure 17: Second Optimisation

When the optimisation finishes, we get results that are very close to the correct policy which is a linear target escapement policy with slope 1.0, Figure 18:



Search: 1/1 Iteration: Criteria:

Search: 1 - Criteria: 58,2552657348831 MrdNOK

Selection criteria:

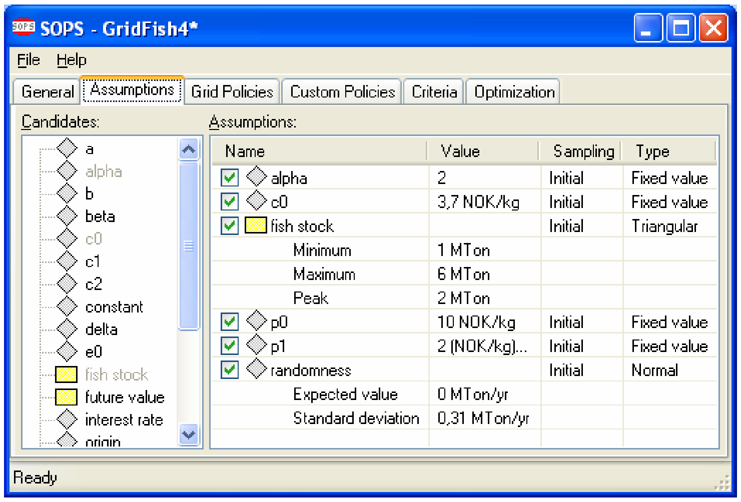
Policy: harvest policy

fish stock (MTon)	
1	-0,336732562513956
2	-0,140342746698309
3	0,860530533042954
4	1,85875515917413
5	1,01607987308807

Ready

Figure 18: Second Optimisation Results

Finally we perform a test with different model assumptions. We change model constants such that the fish price and variable unit costs both vary with the harvest rate. This implies that the fish price declines if the harvest increases, and the unit costs increase when capacity utilisation increases. To achieve this, we change the assumptions in Figure 19.



SOPS - GridFish4*

File Help

General | Assumptions | Grid Policies | Custom Policies | Criteria | Optimization

Candidates:

a, alpha, b, beta, c0, c1, c2, constant, delta, e0, fish stock, future value, interest rate, origin

Assumptions:

Name	Value	Sampling	Type
✓ alpha	2	Initial	Fixed value
✓ c0	3,7 NOK/kg	Initial	Fixed value
✓ fish stock		Initial	Triangular
Minimum	1 MTon		
Maximum	6 MTon		
Peak	2 MTon		
✓ p0	10 NOK/kg	Initial	Fixed value
✓ p1	2 (NOK/kg)...	Initial	Fixed value
✓ randomness		Initial	Normal
Expected value	0 MTon/yr		
Standard deviation	0,31 MTon/yr		

Ready

Figure 19: Assumptions for Variable Price Test

This time we also make a joint optimisation of the constant fishing capacity, e_0. The results in Figure 20 show that harvest for variable fish prices and unit costs is higher at low stocks and it is lower at high stocks than in the previous case with constant fish price and costs. In this case it also makes sense to activate the custom policy for fishing capacity, e_0, and the value is also reported in the table.

	Policy parameters	
	Constant price/costs	Variable price/costs
θ1	-0.34	-0.14
θ2	-0.13	0.35
θ3	0.86	0.73
θ4	1.82	1.06
θ5	1.02	1.32
e0	-	0.41

Figure 20: Policy Parameters for Two Different Models

Finally note that the policy parameters can be easily copied to for instance Excel for further analysis or to make tables like in Figure 20 and graphs like in Figure 21. The graph portrays the data in the table. Clearly, the harvesting policy is sensitive to the assumptions about the price and the costs. Once the results are available, they seem reasonable: one should avoid large harvests with low fish prices and high unit costs. Before the results are available, however, it seems difficult to have a clear idea about strong the effect on the policy should be.

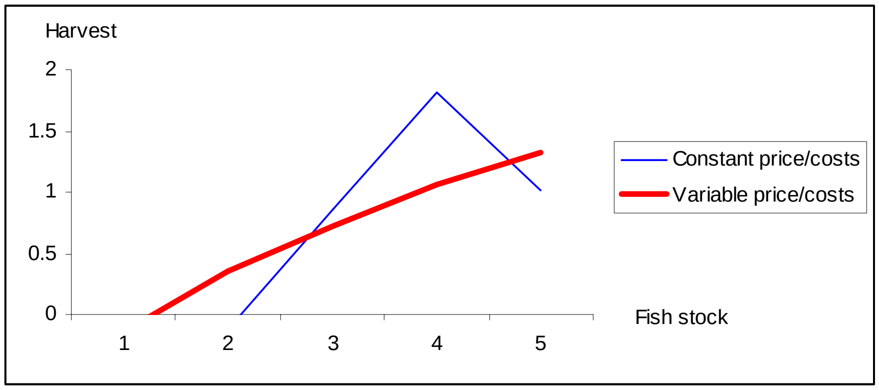


Figure 21: Harvesting policies

Parameters can also be copied from SOPS to Powersim Studio for further analysis there.

4. CONCLUSIONS

We conclude that we have succeeded in producing a very user friendly program for stochastic dynamic optimisation. Testing thus far shows that SOPS reproduces well known solutions to simple problems. Solutions have been found for models with more than ten stock variables and with more than 100 policy parameters in pilot versions of the program. Further testing will help clarify limitations of the program and possibly lead to further improvements.

ACKNOWLEDGEMENTS

The project was financed by the Norwegian Research Council under the program “Bioproduksjon og foredling”. Thanks to Powersim Software AS for programming inputs.

REFERENCES

Kahneman, D., and Tversky, A. (1979). “Prospect Theory: An Analysis of Decision under Risk.” *Econometrica* **47**(2, March 1979):263-291.

Lubow, B.C. (1995). “SDP: Generalized software for solving stochastic dynamic optimization problems.” *Wildlife Society Bulletin* **23**(4):738-742.

Moxnes, E. (2003). “Uncertain measurements of renewable resources: Approximations, harvest policies, and value of accuracy.” *Journal of Environmental Economics and Management* **45**(1):85-108.

Moxnes, E. (2004). “Misperceptions of basic dynamics, the case of renewable resource management.” *System Dynamics Review* **20**(2):139-162.

Moxnes, E. (2005). “Policy Sensitivity Analysis: simple versus complex fishery models.” *System Dynamics Review* ((fortcoming)).

Sterman, J.D. (1989). “Misperceptions of Feedback in Dynamic Decision Making.” *Organizational Behavior and Human Decision Processes* **43**(3):301-335.

Tversky, A., and Kahneman, D. (1974). “Judgment under Uncertainty: Heuristics and Biases.” *Science* **185**:1124-1131.