

Economic Instruction

In this section, the *Journal of Economic Education* publishes articles, notes, and communications describing innovations in pedagogy, hardware, materials, and methods for treating traditional subject matter. Issues involving the way economics is taught are emphasized.

MICHAEL WATTS, Section Editor

Solving Continuous-Time Optimal-Control Problems with a Spreadsheet

Eric Nævdal

Abstract: The author explains how optimal-control problems can be solved with a common spreadsheet such as Microsoft Excel. He illustrates the method with several examples ranging from simple models to quite advanced topics. The method is intended to be beneficial to students and teachers working with complicated theory in the classroom as well as researchers needing a tool for finding numerical solutions to optimal control problems.

Key words: numerical methods, optimal control, shooting

JEL codes: A23, C61, C63

Optimal control (OC) is one the most common techniques in dynamic optimization. Graduate students specializing in macroeconomics or resource economics are usually required to learn OC. Unfortunately, OC is considered by many to be a complicated subject. Also, most teachers emphasize analytical techniques that are of limited use because closed-form solutions to OC problems are rare. In this article, I demonstrate how to solve a variety of continuous-time OC problems using standard features in a spreadsheet program, which is convenient for the following of reasons:

- Students who lack analytical math skills but are familiar with spreadsheets, using simple numerical techniques, can solve problems and gain intuition until their analytical skills catch up.

Eric Nævdal is a research fellow, Department of Forest Sciences, at the Agricultural University of Norway (e-mail: eric.naevdal@isf.nlh.no). The author is grateful for comments from Suzanne Becker, William Becker, Jon M. Conrad, Roberto Garcia, and three anonymous reviewers. This article was partly written while the author was a research fellow at the Department of Economics and Social Sciences, Agricultural University of Norway and visiting at the Department of Economics and Finance, University of Wyoming, whose hospitality is gratefully acknowledged.

- The spreadsheet solution algorithm closely mimics the analytical solution algorithm. Thus working with a spreadsheet solution furthers the students' grasp of the analytical technique.
- This method introduces students to numerical techniques without the need for mastering complicated programming languages like GAMS and Matlab.
- Spreadsheets have many easy-to-use features that allow a visual presentation of control paths. These presentations provide the student with intuition about how OC works and are particularly useful when teaching comparative dynamics.

This article is primarily aimed at instructors who teach OC and want to complement their teaching through simple numerical analysis. I developed the method presented here as a graduate student, and I feel that I gained considerable insights into the mechanics of OC as a result. Quite a few of my fellow students who asked that I show them the method indicated that their understanding of OC was improved by experimenting with numerical solutions. Some academic researchers may also find it useful to learn a simple way of solving simple OC problems without any knowledge of low-level programming. The insights that I gained into numerical methods have led me to tackle more complicated models with Matlab in my postdoctoral research.

The feature in Excel that enables me to solve dynamic optimization problems is the Solver. This tool is very versatile and can be set up to solve almost any optimization problem of the form $\max f(x)$. The Solver also handles constraints and integer problems. The feature I use is the Solver's ability to solve nonlinear systems of equations. The Solver's functionality and flexibility make it a valuable pedagogical tool that should be explored by all economics instructors. A list of papers using Excel to teach economics is maintained by LTSN Economics and can be found at <http://econltsn.ilrt.bris.ac.uk/advice/spreadsheets.htm>. Spreadsheets have been used recently as pedagogical tools in dynamic optimization. Conrad (1999) uses Excel to solve numerous dynamic programming models, and Shone (1997) shows how *discrete* OC problems can be solved. Building on these authors' work, I demonstrate how to use the spreadsheet to solve *continuous*-time problems. Theoretical models found in the literature and textbooks are often framed in a continuous-time framework.

I have assumed that the reader is familiar with OC and the application of Pontryagin's Maximum Principle. The reader who is interested in gaining such familiarity is directed to Seierstad and Sydsæter (1987). The spreadsheets discussed below are available from the author on request.

NUMERICAL SOLUTIONS TO DIFFERENTIAL EQUATIONS

Before students can start solving OC problems with a spreadsheet, they need to know a numerical technique for solving ordinary differential equations (ODEs). Elementary textbooks on differential equations provide a number of algorithms to use when solving numerically ordinary differential equations of the type $\dot{x} = f(t, x)$, $x(0) = x^0$. Borrelli and Coleman (1987) provide a good introduction to the subject. The algorithm that seems to strike the best balance between ease of use, efficiency, and accuracy, and therefore seems to be the most popular,

is the fourth-order Runge-Kutta method. This method is a simple iterative procedure that starts with initial conditions $t = 0$ and $x(0) = x^0$, inserts the initial condition into the algorithm, obtains an estimate for $x(t + \epsilon)$, uses this estimate to obtain an estimate of $x(t + 2\epsilon)$ and so on. ϵ is a step size that must be chosen by the programmer. The formula for obtaining an estimate for $x(t + \epsilon)$ is given by

$$x_{t+\epsilon} = x_t + \epsilon/6(k_1 + 2k_2 + 2k_3 + k_4), \quad (1)$$

where k_i is given by

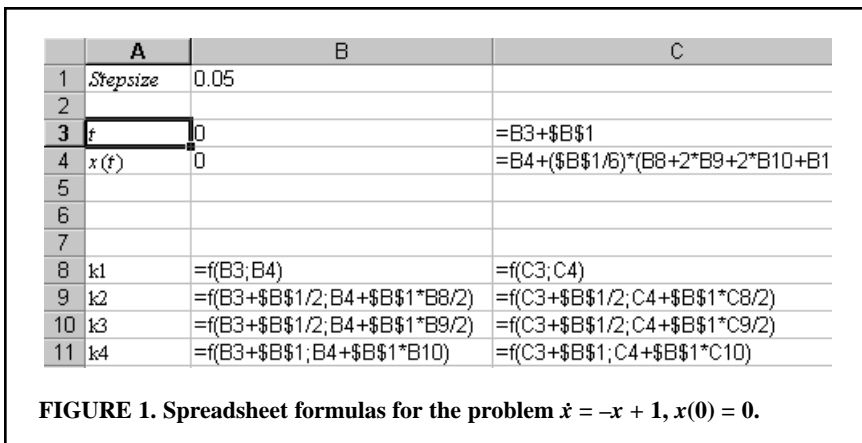
$$\begin{aligned} k_1 &= f(t, x_t) & k_2 &= f(t + \epsilon/2, x_t + k_1\epsilon/2) \\ k_3 &= f(t + \epsilon/2, x_t + k_2\epsilon/2) & k_4 &= f(t + \epsilon, x_t + k_3\epsilon). \end{aligned} \quad (2)$$

Here, x may be a vector or a scalar. To illustrate its use, I solved the differential equation $\dot{x} = -x + 1$ with the initial condition $x(0) = 0$. This equation has the explicit solution $x(t) = 1 - e^{-t}$. First, I write the differential equation as a user-defined function (UDF) in Excel's macro language Visual Basic for Applications (VBA). A short demonstration of how to write such functions is given in the appendix. The written code should look something like this:

```
Function f(t,x)
f = -x + 1
End Function
```

Note that even if f takes t as an argument, t does not appear anywhere in the definition of the function, which allows for generality. When I enter the formulas from equations (1) and (2) into the spreadsheet, using a value of $\epsilon = 0.05$, I get formulas that look like those in Figure 1.¹

In cell B1, I entered the step size, ϵ . In cell B3, I entered the starting point $t = 0$, and in B4, I entered $x(0)$. In B8 to B11, I entered the definitions of k_i from equation (2). Note how references to the step size in B2 have \$ signs, signifying an absolute reference. This ensures that when the content of a cell is copied to another cell, the formula in the new cell still refers to B2. In cell C3, a formula increases time by the step size. In C4, I entered the formula from equation (1).



	A	B	C	D	E	F	G	H	I	J	K	L
1	Stepsize	0.050										
2												
3	f	0.000	0.050	0.100	0.150	0.200	0.250	0.300	0.350	0.400	0.450	0.500
4	x(t)	0.000	0.049	0.095	0.139	0.181	0.221	0.259	0.295	0.330	0.362	0.393
5	x(t) actual	0.000	0.049	0.095	0.139	0.181	0.221	0.259	0.295	0.330	0.362	0.393
6												
7												
8	k1	1.000	0.951	0.905	0.861	0.819	0.779	0.741	0.705	0.670	0.638	0.607
9	k2	0.975	0.927	0.882	0.839	0.798	0.759	0.722	0.687	0.654	0.622	0.591
10	k3	0.976	0.928	0.883	0.840	0.799	0.760	0.723	0.688	0.654	0.622	0.592
11	k4	0.951	0.905	0.861	0.819	0.779	0.741	0.705	0.670	0.638	0.607	0.577
12												

FIGURE 2. Results for the problem $\dot{x} = -x + 1$, $x(0) = 0$.

No more code has to be entered manually. I copied the contents in cells B8–B11 into cells C8–C11, D8–D11, and as far along as necessary. The same was done with the contents of C3–C4. For 10 iterations and with step size 0.05, the results will look like Figure 2.

Note that in row 5 of Figure 2, the explicit solution $x(t) = 1 - e^{-t}$ is given and that there are no discrepancies. For well-behaved ODEs, in particular those that do not have unstable nodes, the Runge-Kutta algorithm is remarkably robust. If the solutions exhibit chaotic or stiff behavior, the method will not work very well. However, this is unlikely to be the case in most OC problems. I discuss another source of instability in the context of OC problems later.

Using Runge-Kutta Methods to Solve Optimal-Control Problems

A standard OC problem is

$$\begin{aligned} \max_u \int_0^T f(t, x, u) dt \\ \text{s.t. : } \dot{x} = g(t, x, u), \quad x(0) = x^0. \end{aligned} \tag{3}$$

The maximum principle gives the information required to solve such problems. The maximum principle is usually stated in terms of the Hamiltonian defined by $H(t, x, p, u) = f(t, x, u) + pg(t, x, u)$. Apart from the transversality conditions, the maximum principle yields

$$u(t, x, p) = \arg \max_y [H(t, x, p, y)] \tag{4}$$

$$\dot{p} = -\frac{\partial}{\partial x} [H(t, x, p, y)] \tag{5}$$

$$\dot{x} = f[t, x, u(t, x, p)]. \tag{6}$$

After inserting for u from equation (4) into equations (5) and (6), I have two differential equations. When solving these equations, I must take into account that the differential equations do not pose a standard initial value problem but a *boundary value* problem. Depending on whether there are equality restrictions on $x(T)$, there is either a transversality condition $p(T) = 0$ or a condition $x(T) = \bar{x}$. In

either case, the value of $p(0)$ is not known. This problem may be solved with a method called *Shooting*. What the method does in the context of OC is to start out with the value $x(0)$ and a more or less informed guess on $p(0)$. Suppose that I want to set $p(T) = 0$. Then starting with an initial guess $p_1(0)$, I go through the Runge-Kutta iterations until I hit T . If $p_1[T|p_1(0)] > 0$, I decrease my guess of $p(0)$. If $p_1[T|p_1(0)] < 0$, I enter a new guess with $p_2(0) < p_1(0)$. There are numerous ways to set up such an algorithm. A Grid Search Algorithm seems to be the most reliable, if not the most efficient from a programmer's perspective. A major feature of this method is that I do not have to bother about this step, because I am going to let Excel take care of it.

SOLVING OPTIMAL-CONTROL PROBLEMS

This section solves four problems, all with a fixed end point, T . These problems are similar to the types of problems first encountered by students of OC.

Problem 1. A Basic OC Problem

Consider the following problem:

$$\max_u \int_0^1 \left(x - \frac{1}{2} u^2 \right) dt \quad (7)$$

s.t. : $\dot{x} = u - x$, $x(0) = 0$ $x(1)$ free.

Equation 7 may be considered a stylized model of an investment process, where x is real capital, depreciating at rate 1, and u is investment. Revenues depend on the level of capital, and costs depend on investment. To solve this problem, I proceed with the following steps:

1. Find the $u(t, x, p)$ that solves equation (4) in the present case. It turns out that for the model in equation (7),

$$u(t, x, p) = p.$$

2. Insert for $u(t, x, p)$ into equations (5) and (6). This gives

$$\dot{p} = -1 + p \text{ and } \dot{x} = p - x.$$

3. Write the expressions from number (2) as functions in Excel's macro language. The functions should look something like this:

```
Function dP(t, x, p)
dP = -1 + p
End Function
Function F(t, x, p)
F = p - x
End Function
```

The first function is the differential equation for the costate variable p ; the second is the differential equation for the state variable x . (The superfluous arguments are shown for generality and to make the code easy to reuse.)

- In cell A2, enter a small number for the step size. As a general rule, some number in the range 0.01 to 0.05 works well.
- Enter the rows that will show the state variable and the costate variable. The first columns should look like Figure 3.

The first row in Figure 3 is the variable t . It is updated by the step size given in cell A2. The second row is x ; the third row is p . Note how the formulas in the second and third rows match the Runge-Kutta formula in equation (1). The number in C3 is $p(0)$. This is the unknown of the problem. The initial guess here is 0, but any number may be chosen.

- Write formulas for k_1 , k_2 , k_3 , and k_4 into the spreadsheet. These should look something like Figure 4.

Note the column and row headers. The only thing to write here is the first column in cells C7 to C14. I can then copy the contents of this column to cells D7 to D14, E7 to E14, all the way up to cells IV7 to IV14, if so desired. In this example, I copied the contents of C7 to C14 into the cells D7–D14 up to W7–W14. The content of cell C7 is k_1 for the variable x at time $t = 0$. In this example, it is simply $p(0) - x(0)$; the content of cell C8 is k_2 for the variable x ; C9 is k_3 for x , and C10 is k_4 . Then at C11, I have k_1 for the variable p and so on.

Steps 5 and 6 are the most complicated parts, and they require no more than five minutes of typing. Also, after having done it once, the same sheet may be used again for any problem with one state variable.² For typographical reasons, the cells have been illustrated with time going from left to right. Because Excel has 65,336 rows but only 256 columns, it is beneficial to let time go from top to bottom in problems where final time is relatively large. If the reader chooses to

	B	C	D	E
1	t	0	=C1+\$A\$2	=D1+\$A\$2
2	x	0	=C2+(\$A\$2/6)*(C7+2*C8+2*C9+C10)	=D2+(\$A\$2/6)*(D7+2*D8+2*D9+D10)
3	p	0	=C3+(\$A\$2/6)*(C11+2*C12+2*C13+C14)	=D3+(\$A\$2/6)*(D11+2*D12+2*D13+D14)

FIGURE 3. Spreadsheet formulas for problem 1.

	B	C	D
6			
7	x	$k_1 = F(C\$1;C\$2;C\$3)$	$=F(D\$1;D\$2;D\$3)$
8	x	$k_2 = F(C\$1+ \$A\$2/2;C\$2+0.5* \$A\$2*C7;C\$3+0.5* \$A\$2*C11)$	$=F(D\$1+ \$A\$2/2;D\$2+0.5* \$A\$2*D7;D\$3+0.5* \$A\$2*D11)$
9	x	$k_3 = F(C\$1+ \$A\$2/2;C\$2+0.5* \$A\$2*C8;C\$3+0.5* \$A\$2*C12)$	$=F(D\$1+ \$A\$2/2;D\$2+0.5* \$A\$2*D8;D\$3+0.5* \$A\$2*D12)$
10	x	$k_4 = F(C\$1+ \$A\$2;C\$2+ \$A\$2*C9;C\$3+ \$A\$2*C13)$	$=F(D\$1+ \$A\$2;D\$2+ \$A\$2*D9;D\$3+ \$A\$2*D13)$
11	p	$k_1 = dP(C\$1;C\$2;C\$3)$	$=dP(D\$1;D\$2;D\$3)$
12	p	$k_2 = dP(C\$1+ \$A\$2/2;C\$2+0.5* \$A\$2*C7;C\$3+0.5* \$A\$2*C11)$	$=dP(D\$1+ \$A\$2/2;D\$2+0.5* \$A\$2*D7;D\$3+0.5* \$A\$2*D11)$
13	p	$k_3 = dP(C\$1+ \$A\$2/2;C\$2+0.5* \$A\$2*C8;C\$3+0.5* \$A\$2*C12)$	$=dP(D\$1+ \$A\$2/2;D\$2+0.5* \$A\$2*D8;D\$3+0.5* \$A\$2*D12)$
14	p	$k_4 = dP(C\$1+ \$A\$2;C\$2+ \$A\$2*C9;C\$3+ \$A\$2*C13)$	$=dP(D\$1+ \$A\$2;D\$2+ \$A\$2*D9;D\$3+ \$A\$2*D13)$

FIGURE 4. Spreadsheet formulas for problem 1—continued.

type the formulas as in Figures 1–4, this can easily be achieved by copying all the cells with formulas and text, opening a new sheet, and doing a Paste Special, taking care to mark the check box Transpose. This is done in the sequel, and time hereafter goes from top to bottom.

7. The final step is to use the Solver application to solve the problem. Prior to solving the problem, the table of t , x , and p values looks like Figure 5.

The initial value of $x(0)$ is correct. However, because $p(0)$ is arbitrarily set at 0, the transversality condition $p(1)$ is incorrect, as are the paths of x and p . Now load the Solver application in Excel; set the Solver up so that it sets the contents of cell C24 to zero by changing cell C4. The Solver box should look like Figure 6. Pressing Solve provides the optimal paths of x and p . These appear approximately as in columns B and C in Figure 7.

It is easy to verify that the exact solution to this problem is

$$x(t) = 1 - \frac{1}{2}e^{t-1} + (\frac{1}{2}e^{-1} - 1)e^{-t}, \text{ and } p(t) = 1 - e^{t-1}.$$

The exact values are given in columns D and E. As seen in Figure 7, the discrepancies are minute. Note that even if the numerical solution does not give $p(T) = 0$, the difference is negligible.

	A	B	C
2	Stepsize:	0.05	
3	t	x(t)	p(t)
4	0	0	0
5	0.05	-0.00125	-0.05127
6	0.1	-0.005	-0.10517
7	0.15	-0.01127	-0.16183
8	0.2	-0.02007	-0.2214
9	0.25	-0.03141	-0.28403
10	0.3	-0.04534	-0.34986
11	0.35	-0.06188	-0.41907
12	0.4	-0.08107	-0.49182
13	0.45	-0.10297	-0.56831
14	0.5	-0.12763	-0.64872
15	0.55	-0.1551	-0.73325
16	0.6	-0.18547	-0.82212
17	0.65	-0.2188	-0.91554
18	0.7	-0.25517	-1.01375
19	0.75	-0.29469	-1.117
20	0.8	-0.33744	-1.22554
21	0.85	-0.38354	-1.33965
22	0.9	-0.43309	-1.4596
23	0.95	-0.48623	-1.58571
24	1	-0.54309	-1.71828

FIGURE 5. Initial (nonoptimal) paths for problem 1.

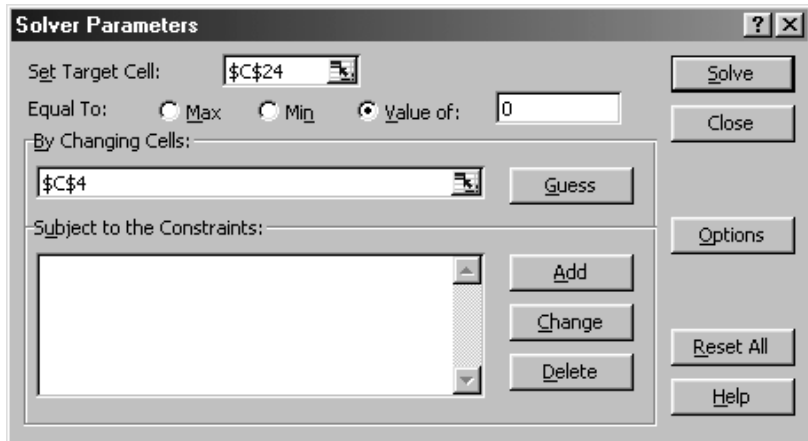


FIGURE 6. Solver options for problem 1.

	A	B	C	D	E
2	Stepsize	0.05			
3	f	x	p	Exact x	Exact p
4	0	0	0.63212054	0	0.632121
5	0.05	0.030368936	0.613258958	0.030369	0.613259
6	0.1	0.058313289	0.593430322	0.058313	0.59343
7	0.15	0.083902935	0.57258505	0.083903	0.572585
8	0.2	0.107201862	0.550671018	0.107202	0.550671
9	0.25	0.128268329	0.52763343	0.128268	0.527633
10	0.3	0.147155012	0.503414679	0.147155	0.503415
11	0.35	0.163909139	0.477954206	0.163909	0.477954
12	0.4	0.178572604	0.451188347	0.178573	0.451188
13	0.45	0.191182072	0.423050174	0.191182	0.42305
14	0.5	0.201769075	0.393469325	0.201769	0.393469
15	0.55	0.210360085	0.362371834	0.21036	0.362372
16	0.6	0.216976584	0.329679941	0.216977	0.32968
17	0.65	0.221635116	0.295311898	0.221635	0.295312
18	0.7	0.224347332	0.259181768	0.224347	0.259182
19	0.75	0.225120011	0.221199207	0.22512	0.221199
20	0.8	0.223955088	0.181269239	0.223955	0.181269
21	0.85	0.220849648	0.139292017	0.22085	0.139292
22	0.9	0.215795926	0.095162577	0.215796	0.095163
23	0.95	0.208781286	0.048770573	0.208781	0.048771
24	1	0.199788188	1.96163E-14	0.199788	0

FIGURE 7. Optimal paths for problem 1.

Problem 2. End-Point Constrained-State Variable

This problem is a simple modification of problem 1. The problem to be solved is

$$\max_u \int_0^1 \left(2x - \frac{1}{2}u^2 \right) dx \quad (8)$$

$$s.t. : \dot{x} = u - x, \quad x(0) = 0 \quad x(1) = 0.$$

Instead of letting $x(1)$ be free, it is now required that $x(1) = 0$. Also, the criterion has changed slightly. The modifications to the spreadsheet needed to change problem 1 into this problem are minor. The Runge-Kutta algorithm is written into the spreadsheet for general functions of \dot{x} and \dot{p} , so I only have to change the functional forms. In this case, $\dot{p} = -2 + p$ and $\dot{x} = p - x$. The defined functions should appear as:

```
Function dP(t,x,p)
dP = -2 + p
End Function
Function F(t,x,p)
F = p - x
End Function
```

After changing the code, the only thing that remains to be done is to change the endpoint conditions. Problem 1 had no restrictions on $x(1)$, so $p(1) = 0$. In problem 2, $x(1) = 0$ is imposed, so there are no restrictions on $p(1)$. I change this in the Solver box, by changing the *Set Target Cell* from $\$C\24 to $\$B\24 . The Solver box should look like Figure 8.

Having done that, click Solve, and the problem is solved. The results are given in Figure 9.

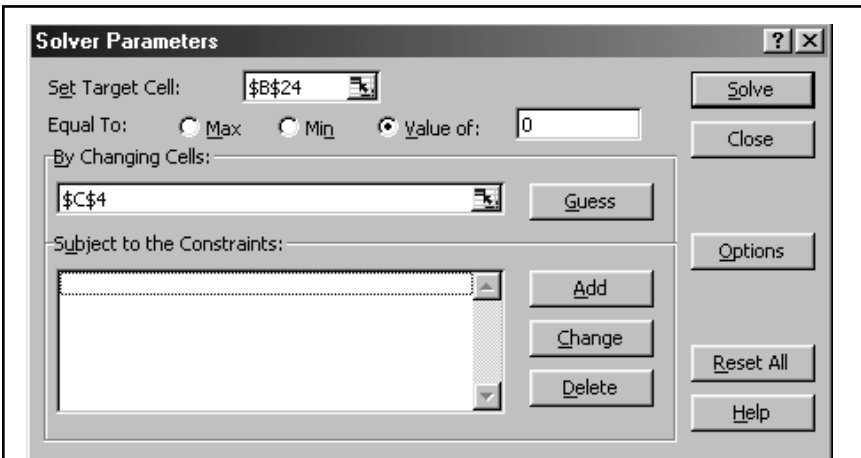


FIGURE 8. Solver options for problem 2.

	A	B	C
3	t	x	p
4	0	0	0.924234277
5	0.05	0.043730448	0.869078592
6	0.1	0.082569203	0.811095014
7	0.15	0.116613384	0.750138555
8	0.2	0.145948117	0.686056792
9	0.25	0.170646755	0.618689487
10	0.3	0.190771058	0.547868186
11	0.35	0.206371347	0.4734158
12	0.4	0.21748663	0.395146158
13	0.45	0.224144702	0.312863546
14	0.5	0.226362211	0.226362215
15	0.55	0.224144702	0.135425866
16	0.6	0.21748663	0.03982711
17	0.65	0.206371347	-0.060673098
18	0.7	0.190771058	-0.166326061
19	0.75	0.170646756	-0.277395968
20	0.8	0.145948117	-0.39416055
21	0.85	0.116613384	-0.51691178
22	0.9	0.082569204	-0.6459566
23	0.95	0.043730448	-0.781617689
24	1	4.89192E-14	-0.92423427

FIGURE 9. Optimal paths for problem 2.

The value for $x(1)$ for all practical purposes is equal to 0, which is what was required.³ The path of $p(t)$ shows positive values until $t = 0.65$ and negative values thereafter. There is economic sense to this. The interpretation of $p(t)$ is that it is the shadow price on x at time t . Because x contributes positively to instantaneous utility, it is valuable to begin with. However, it is costly to reduce x , so the requirement that $x(1) = 0$ implies that as time gets close to 1, one would rather have less of it.

Problem 3. A Bang-Bang Problem

Problem 2 is now modified to be linear in the control and should be studied as

$$\max_{u \in [0,1]} \int_0^1 \left(2x - \frac{1}{2}u \right) dt \tag{9}$$

s.t. : $\dot{x} = u - x$, $x(0) = 0$ $x(1) = \text{free}$.

Such problems lead to *bang-bang* controls, that is, problems where the control jumps from one boundary to another. In this case, u is required to take values in the range $[0, 1]$. Note that $x(1)$ is again allowed to take any value. Here it is beneficial to do a little work with paper and pencil. The control variable u should maximize the Hamiltonian

$$H(t, x, p, u) = 2x - \frac{1}{2}hu + p(u - x).$$

It is straightforward to see that this implies that if $p(t) > \frac{1}{2} \Rightarrow u(t) = 1$ and $p(t) < \frac{1}{2} \Rightarrow u(t) = 0$. The simplest way of incorporating this fact into the problem is to modify the functions defined in Excel. A simple If ..., then ..., else ..., end statement will accomplish this. The functions should look like this:

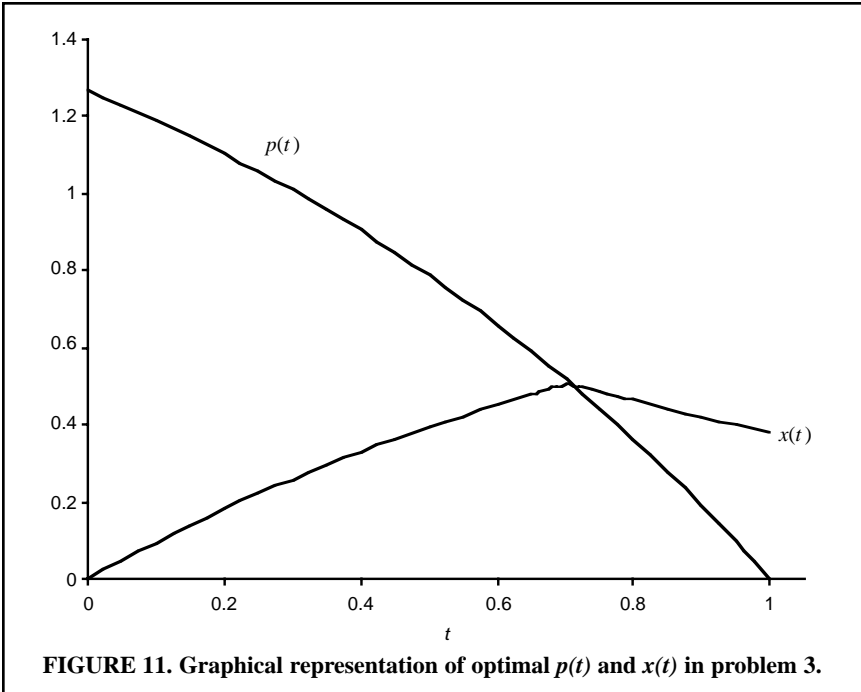
```
Function dP(t,x,p)
dP = -2 + p
End Function
Function F(t,x,p)
  If p > = .5 Then
    u = 1
  Else
    u = 0
  End If
  F = u - x
End Function
```

After doing this modification, the Solver box needs to look like Figure 6 again, because $x(1)$ is free. Then click OK, and the problem is solved. The solution is presented in Figure 10.

One may graph the data using Excel's extensive graphical possibilities. A graph of $t, x(t)$ and $p(t)$ will look like Figure 11. Note that the graph shows a kink in $x(t)$ when $p(t) = \frac{1}{2}$ and that $p(1) = 0$.

	A	B	C	D
3	t	x(t)	p(t)	
4	0	0	1,26424145	u=1
5	0,05	0,0487706	1,2265183	u=1
6	0,1	0,0951626	1,18686105	u=1
7	0,15	0,139292	1,14517053	u=1
8	0,2	0,1812692	1,10134249	u=1
9	0,25	0,2211992	1,05526733	u=1
10	0,3	0,2591818	1,00682985	u=1
11	0,35	0,2953119	0,95590893	u=1
12	0,4	0,3296799	0,90237724	u=1
13	0,45	0,3623718	0,84610092	u=1
14	0,5	0,3934693	0,78693926	u=1
15	0,55	0,4230502	0,72474431	u=1
16	0,6	0,4511883	0,65936055	u=1
17	0,65	0,4779542	0,5906245	u=1
18	0,7	0,5034147	0,51836428	u=1
19	0,75	0,4867897	0,44239919	u=0
20	0,8	0,4630487	0,3625393	u=0
21	0,85	0,4404655	0,27858489	u=0
22	0,9	0,4189838	0,19032606	u=0
23	0,95	0,3985497	0,0975421	u=0
24	1	0,3791122	1E-06	u=0

FIGURE 10. Optimal paths for problem 3.



Problem 4. A Salvage-Value Problem

Salvage-value problems are common in economics. The general form of a salvage-value problem is

$$\begin{aligned} \max_u \int_0^T f(t, x, u) dt + S[T, x(T)] \\ \text{s.t. : } \dot{x} = g(t, x, u), \quad x(0) = x^0. \end{aligned} \tag{10}$$

The maximum principle holds for this class of problems. The only modification is that the transversality condition $p(T) = 0$ must be replaced with $p(T) = S'_x [T, x(T)]$. The necessary modifications to the method are illustrated with the following example:

$$\begin{aligned} \max_u \int_0^1 \left(2x - \frac{1}{2} u^2 \right) dt - [x(1)]^2 \\ \text{s.t. : } \dot{x} = u - x, \quad x(0) = 0, \quad x(1) \text{ free.} \end{aligned} \tag{11}$$

This is the same as problem 1, with the addition of the term, $-[x(1)]^2$, which gives utility at time 1 derived from the stock of x . An interpretation of this term may be clean-up costs associated with the shut down of a plant. The differential equations for \dot{x} and \dot{p} are the same as in problem 2. The transversality condition is $p(1) = -2x(1)$. In Excel, define the following function as the derivative of the salvage value with respect to x :

Function $dS(t, x, p)$

```
dS = -2 * x
End Function
```

After defining the derivative of the salvage value function as a function, enter the expression $p(1) - 2x(1)$ in a cell as in Figure 12. The correct formula is shown in Excel's formula bar. The Excel Solver can now find the solution. Load the Solver and instruct it to set cell C26 to 0 by changing cell C4. The result should look like Figure 13. Note how the co-state develops. First, it is positive, reflecting the state-variables contribution to instantaneous utility. After $t = 0.8$, it is negative reflecting the cost of $x(1)$ because of the negative salvage value.

COMPARATIVE DYNAMICS

Comparative dynamics is one of the most challenging aspects of OC. To grasp the analytical techniques, the student must be proficient in some relatively complicated mathematics. I show now how to visually inspect the effects of parameter changes by using Excel's graphical capabilities. By examining such figures, the student can gain valuable insights about the effect of changes in parameters before acquiring the mathematical skills needed to perform the mathematical analysis. Consider the following generalized version of problem 1

$$\max_u \int_0^1 \left(ax - \frac{1}{2} u^2 \right) dt \quad (12)$$

$$s.t. : \dot{x} = u - x, \quad x(0) = 0, \quad x(1) \text{ free.}$$

The term ax may be thought of as a production function translating the stock of capital into instantaneous production. If I want to examine the effect of letting a vary, I can do so by comparing solutions. The first step is to rewrite the code for the defined functions in problem 1. The differential equation for the

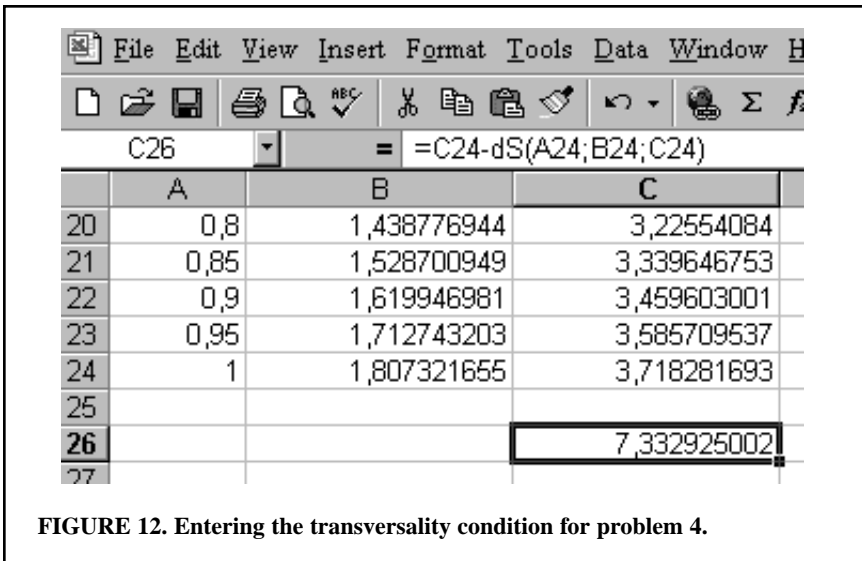


FIGURE 12. Entering the transversality condition for problem 4.

	A	B	C
3	t	x	p
4	0	0	0,553287971
5	0,05	0,026425665	0,530384557
6	0,1	0,050416887	0,506306886
7	0,15	0,072033657	0,480994673
8	0,2	0,091330029	0,454384702
9	0,25	0,108354252	0,426410409
10	0,3	0,123148896	0,397001843
11	0,35	0,135750957	0,366085468
12	0,4	0,146191945	0,333583976
13	0,45	0,154497968	0,299416098
14	0,5	0,160689796	0,263496395
15	0,55	0,164782911	0,22573505
16	0,6	0,166787549	0,186037639
17	0,65	0,166708722	0,144304898
18	0,7	0,164546233	0,100432475
19	0,75	0,160294675	0,054310664
20	0,8	0,153943416	0,005824137
21	0,85	0,145476575	-0,045148347
22	0,9	0,134872981	-0,098734246
23	0,95	0,122106118	-0,155067552
24	1	0,107144065	-0,214289129
25			
26		Transversality condition	-1E-06
27			

FIGURE 13. Optimal paths for problem 4.

co-state in this problem is given by $\dot{p} = a - p$. To make the changes in a easier to implement, I find it beneficial to define a as a constant in Excel's macro language. This is particularly useful with more complex models. The analytical expressions with the appropriate symbols can then be written and easily changed, even if the same symbol is evaluated several times in the calculation. To define a constant, I can use a Const statement.⁴ The code for this problem is then

```
Public Const a as single = 1
Function dP(t,x,p)
dP = -a + p
End Function
Function F(t,x,p)
F = p - x
End Function
```

Solving the problem repeatedly while letting the constant a take the values 0.5, 1, and 2, I get different solutions. These values may be copied to a different spreadsheet and graphed.⁵ In this particular case, the result will look like Figure 14. Figure 14 displays no surprises. The higher the value of a , the higher is $p(t)$ for all t less than 1, and the higher is $x(t)$ for all t larger than 0.

TECHNICAL HINTS AND NOTES FOR EXTENDING THE TOOL

Increasing Computational Speed

Defining functions in Excel's macro language, Visual Basic, has a significantly adverse effect on computational speed. The option exists to enter the required formulas directly in the cells of the spreadsheet. In addition to increasing speed, this has the benefit of avoiding some of the quirks of Excel's macro language. Occasionally, when an UDF function returns an error such as "divide by zero" or the number exceeds the limitations imposed by the double-precision floating-point format, Excel gets stuck (see below) and refuses to recalculate even if the arguments of the function are changed. The file must be closed and reopened before Excel reports a new answer in the cell. The drawback of entering formulas into cells directly is that it requires more work and makes re-use of the spreadsheet for new problems more cumbersome.

Increasing Reliability

When an UDF is given an argument that leads to an error such as "divide by zero," a negative number raised to the power of a negative number, or a number that is too large, the defined functions will return a #VALUE message. This is fatal to the Solver application and makes it stop, which may be a problem, even if along the optimal paths x and p should behave nicely. Two things may go

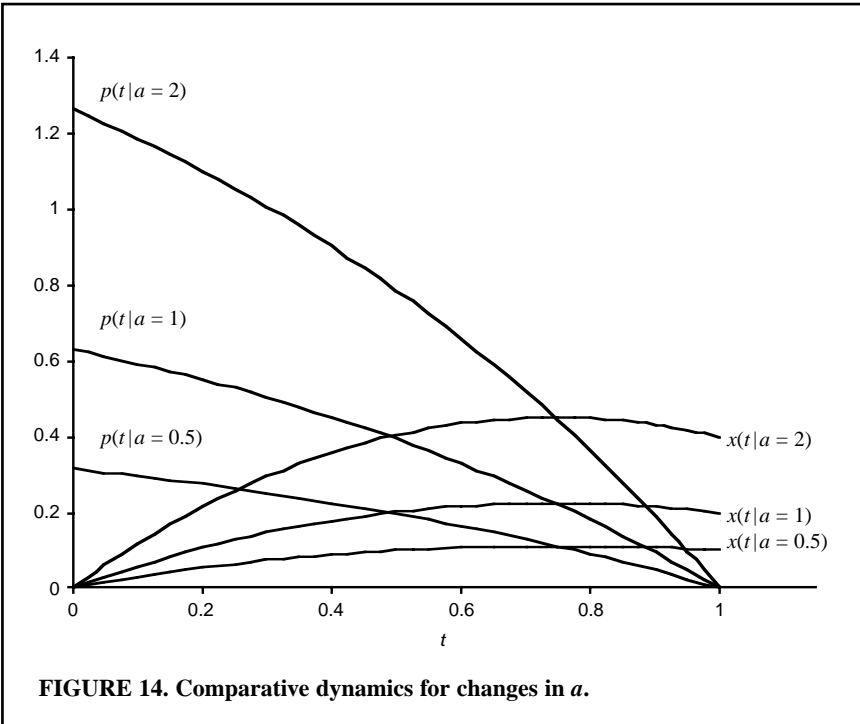


FIGURE 14. Comparative dynamics for changes in a .

wrong. First, if $x(t)$ or $p(t)$ at some point along the optimal path is close to zero, that is, if $x(0) = 0$, then the x argument for k_2 , k_3 , or k_4 may take values that are less than zero. Also, when the Solver looks for an optimal path, it may occasionally make guesses that lead to values that are negative. If the differential equations include terms like $x^{(a-1)}$ or $\ln(x)$ the result will be a #VALUE message, and the Solver does not recover from these errors. One can avoid this problem by using Excel's Goal Seek feature instead of the Solver. Goal Seek uses an algorithm that is sturdier and more forgiving than the Solver. However, it is slower and will only work for problems with one state variable. Also, Goal Seek tends to deliver results that are numerically less accurate, although not significantly so.

Dealing with Saddle Points

Long-run steady states in OC are often saddle points, which are a source of numerical instability. If one tries to use the method presented in the previous section with a time horizon that is too large, this instability may produce strange results. Judd (1999) recommends a method called *Reverse Shooting*, which is straightforward to apply with the technique presented above. The method above makes guesses about $p(0)$ to satisfy the transversality conditions. Reverse Shooting does the exact opposite. One starts out with the correct transversality condition and varies $x(T)$ until $x(0)$ is correct. For this to work, one must alter the Runge-Kutta algorithm so that it works backward instead of forward. The modified algorithm looks like this:

$$x_{t-\varepsilon} = x_t - \varepsilon/6(k_1 + 2k_2 + 2k_3 + k_4) \quad (13)$$

where k_i is given by

$$\begin{aligned} k_1 &= f(t, x_t) & k_2 &= f(t - \varepsilon/2, x_t - k_1\varepsilon/2) \\ k_3 &= f(t - \varepsilon/2, x_t - k_1\varepsilon/2) & k_4 &= f(t + \varepsilon, x_t - \varepsilon k_3). \end{aligned} \quad (14)$$

Be aware, however, that if the steady state is not a saddle point but an attractor, there may be numerical instability if one applies reverse shooting. Straight shooting is then the appropriate algorithm. The stability properties of the steady state should therefore be considered before writing the spreadsheet. For short-time horizons, this issue usually does not arise.

Numerical Optimization of the Hamiltonian

All examples in this article are such that the optimal control as a function of state and costate variables can be found with pencil and paper, and the resulting function $u(t, x, p)$ can be inserted into the expressions for \dot{x} and \dot{p} . Occasionally, this is not possible, and the Hamiltonian must be maximized numerically. In principle, these problems can be solved with Excel. In practice, the increase in the number of computations and Visual Basic's limitations as a programming language implies that this task requires considerable effort. Also, the increase in computational effort is huge. For each Runge-Kutta iteration, one must find val-

ues of u that maximizes the Hamiltonian: once at time t , twice at time $t + \varepsilon/2$, and once at time $t + \varepsilon$. Although the number of iterations needed for each optimization should be small if previously computed values of u are used as guesses, it should be clear that the program execution time increases substantially. Therefore, there are considerable benefits from choosing functional forms to find analytical expressions for $u(x,p)$.

USING NUMERICAL OPTIMAL CONTROL IN THE CLASSROOM

Optimal-control theory is taught in a variety of courses in different programs. Some are plain math courses, typically in dynamic optimization. In such courses, OC theory is usually presented rigorously with proofs of the relevant mathematical theorems. In economic theory courses, OC (and/or its predecessor calculus of variations) is typically taught as a part of resource economics and macroeconomics courses. OC is less often taught in courses in public economics and cost/benefit analysis, although OC surely belongs in such a context. Use of the methods presented here should be tailored to the syllabus of the specific courses. In a rigorous course in dynamic optimization, the instructor may want to derive the Runge-Kutta method explicitly before presenting the shooting algorithm. A rigorous presentation of the fundamentals of Runge-Kutta and other algorithms for the solution of ODEs may be found in Judd (1999).

Economic theory classes will often put less emphasis on methodology tools and only present analytical methods in sufficient depths so that economic insights can be presented. For economic theory instructors who want to use the methods described here, the following approach may be fruitful. First, the instructor presents the results in an analytical framework. Thereafter, the students may be given homework assignments in which they are asked to verify the analytical results with a given parametric representation of the theoretical model.

I present three models below that, in addition to the investment problem described above, are likely to be discussed in different types of courses. Suggestions are given for exercises that present the students with economically interesting challenges that should increase their understanding of the models. I recommend that students be required to code the spreadsheets and set up the Solver themselves rather than being given copies. Considerable insight can be gained from the mental process of setting the problem up right, rather than being given a completed spreadsheet that can be used as a black box for crunching out numbers.

Economic Growth—The Solow Model

The first encounter many students have with dynamic analysis is Solow's model of economic growth, which is not an optimal control model but leads to an ordinary differential equation. However, as an introduction to dynamic optimization, I often derive the golden rule of capital accumulation. The model is developed from the following three equations:

$$X = F(K,L), \quad (15)$$

$$\dot{K} = sX \quad K(0) = K_0, \quad (16)$$

$$\dot{L} = \lambda L \quad L(0) = L_0, \quad (17)$$

where X is aggregate production, determined by the inputs capital, K , and labor, L . $F(K,L)$ is a production function assumed to be homogeneous of degree 1. The savings rate is s , and the relative growth in the labor force is λ . Equations (15), (16), and (17) are used to derive the following differential equation:

$$\dot{k} = sf(k) - \lambda k \quad k(0) = k_0 = K_0/L_0, \quad (18)$$

The variable k is given by K/L and $f(k) = f(K/L) = F(K/L, 1)$. If $F(K,L)$ is the Cobb-Douglas production function $AK^\alpha L^{1-\alpha}$, then $f(k) = Ak^\alpha$. Equation (18) may then be written

$$\dot{k} = sAk^\alpha - \lambda k \quad k(0) = k_0 = K_0/L_0, \quad (19)$$

This differential equation can be solved using the Runge-Kutta method presented above with no need to use the Solver. However, the Solver can be used to solve for the savings rate that maximizes the steady-state level of consumption, the golden rule of capital accumulation. Analytically, one can find this expression by solving

$$\max_{s,k} (1-s)f(k) \text{ s.t.: } sf(k) - \lambda k = 0. \quad (20)$$

Solving problems like equation (20) is a standard feature of the Excel Solver and is not discussed further here. The golden rule value of s can then be incorporated into the Runge-Kutta algorithm, and consumption paths can be compared with consumption paths using exogenous values of s .

Fisheries

The economics of fisheries is an important part of most courses on natural resource economics. The standard model of fisheries is the Shaeffer harvesting function. A good exposition of the economics of fisheries using the Shaeffer function is found in Clark (1976). This function is given by

$$\dot{x} = x(1 - x/K) - qxE. \quad (21)$$

Here x is the stock of fish, K is the carrying capacity or the steady-state stock of fish in the absence of harvesting, q is the “catchability” coefficient, and E is the harvesting effort. Assuming that π is the constant price of fish and that harvesting cost is a quadratic function of effort gives rise to the following model:

$$\max_E \int_0^T (\pi qx E - cE^2) e^{-rt} dt, \quad (22)$$

subject to equation (21) and $E \geq 0$. An analytical solution for this kind of model does not exist. Maximizing the Hamiltonian with respect to E gives the optimal harvest rate as a function of x and the co-state p . E is given by $E = (2c)^{-1}(\pi -$

$pe^{rt}qx$. Coding the differential equations for the state and the costate variables into Excel can be done as follows:

```
Function f(t,x,p)
E = [(q * x)/c] * [Pi - p * Exp(r * t)]
f = x * [1 - (x/K)] - q * x * E
End Function
Function dp(t,x,p)
E = [(q * x)/c] * [Pi - p * Exp(r * t)]
dp = -Pi * q * E * Exp(-r * t) + p * q * E - p * [1 -
(2 * x/K)]
End Function
```

The constants must be defined with Public Const as shown above. Note that the price of fish, π , is coded as *Pi*. The fisheries model is highly nonlinear and the steady state is a saddle point. If the Solver is used, it will often result in #VALUE messages as described earlier. One should use the Goal Seek application in Excel when solving this model. The students should be asked to examine comparative dynamics. Also, it is straightforward to model open access fisheries. Open-access fisheries are the most useful benchmark with which to compare optimal policies. Open access without friction implies that profits should be 0 at all times. This implies that $\pi qx E - cE^2 = 0 \Rightarrow E = \pi qx/c$ for $x > 0$. The differential equation for the stock of fish can then be written

$$\dot{x} = x(1 - x/K) - \pi q^2 x^2. \quad (23)$$

Equation (23) can be solved without any further optimization using the Runge-Kutta method. This model assumes that fisheries are able to respond immediately to profit opportunities. An alternative is to assume that one can change effort only gradually, that is, open access with friction, by letting effort be determined by the following differential equation:

$$\dot{E} = \gamma(\pi qx E - cE^2). \quad (24)$$

Here, E will increase if rents are positive and decrease when rents are negative. The rate of change in E is determined by the parameter γ . Equations (21) and (24) form a system of differential equations that may be solved with the Runge-Kutta algorithm. Again, comparison with the optimally regulated fishery is beneficial.

A challenging problem for the students is how to adapt the textbook infinite horizon model to a finite horizon model that can be used in practical regulation. One approach is to calculate the steady state, x_{ss} , in the infinite horizon model. The students can then set up different models in which the finite time model approaches the steady state in various ways. The most obvious way of doing this is to specify an end constraint $x(T) = x_{ss}$. Unfortunately, this is usually incorrect.⁶ One approach is to choose a time horizon so large that x is close to x_{ss} , for a substantial part of the planning horizon and treat the solution as optimal for an initial time period. One way of doing this is to find a time horizon T such that the optimal solution for that time horizon obeys $|x(t) - x_{ss}|/x_{ss} < \delta$ over an interval $[t_1, t_2]$, where $t_2 < T$. δ is some sufficiently small number, and $[t_1, t_2]$ should be

chosen so that $t_2 - t_1 > 1/3T$. Then let t^* be defined in the following way: If $x(0) < x_{ss}$, then $x'(t^*) = 0$ for some $t^* \in [t_1, t_2]$ or if $x(0) > x_{ss}$ then $x''(t^*) = 0$ for $t^* \in [t_1, t_2]$ and let the model represent optimal planning in $[0, t^*]$.

The idea is illustrated in Figure 15. The fisheries model is solved for parameter values

$$x(0) = 5, q = 1, \pi = 1, r = 0.4, c = 2, K = 20, \text{ and } T = 13.$$

Note how $x(t)$ is almost constant in the interval $[t_1, t_2] = [3, 8]$. In this interval, the behavior of the optimal solution closely mimics the steady-state behavior. At $t = t^* = 6.8$, the derivative of the steady-state variable changes sign. Until t^* , the system gets closer and closer to the steady state. After t^* , the system goes further and further away, although initially quite slowly. The optimal path of this finite time model will be almost identical to the infinite horizon model until $t = 6.8$.

Consumption and Saving

One of the first applications of dynamic optimization in economics was the Ramsey model of consumption and savings (Blanchard and Fisher 1989). Although originally phrased as a calculus-of-variations problem, it is straightforward to phrase it as an optimal-control problem. Assuming that utility is logarithmic, the standard model is

$$\max_c \int_0^T \ln(c) e^{-\rho t} dt \tag{25}$$

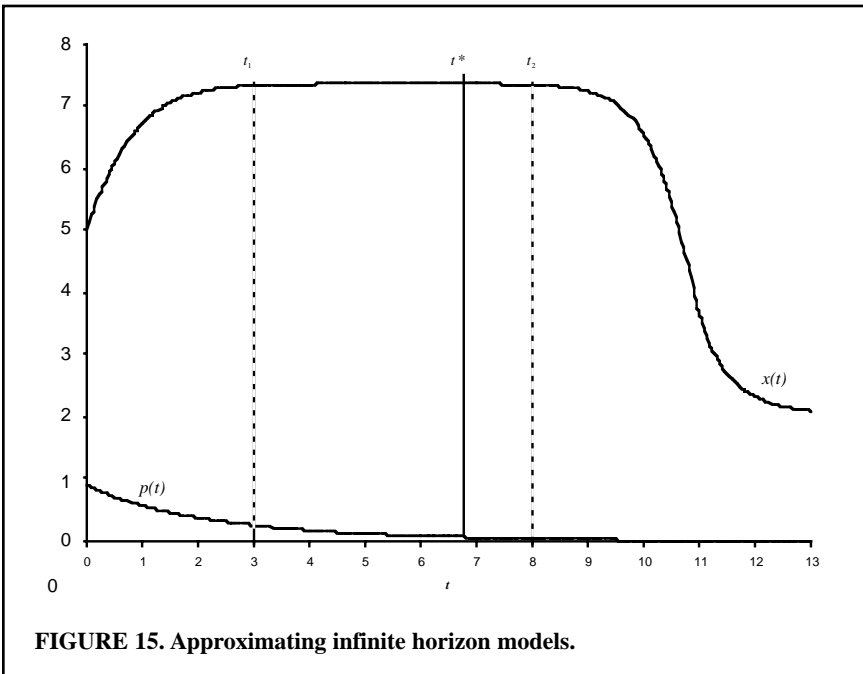


FIGURE 15. Approximating infinite horizon models.

subject to $\dot{x} = w(t) + rx - c$, $x(0) = x^0$, and $x(T) \geq 0$. Here, x is wealth, c is consumption, ρ the subjective rate of discounting, $w(t)$ is instantaneous income/wage rate, x is wealth, and r is the market interest rate. T can be normalized to unity. Some familiar theorems may be illustrated in this model. The students may be asked to show the following results:

1. By keeping r fixed and letting ρ vary, one can show that if $r = \rho$, then consumption will be constant over time whereas $\rho > r$ will lead to consumption patterns skewed towards the present.

2. One can analyze the effect that different wage schedules have on saving throughout the lifespan. For example, if wages peak in the middle of a lifespan and are lower earlier and later, this can illustrate the process of dissaving early and later in life and the accumulation of wealth in middle age. If time is normalized to unity, letting $w(t) = -4t(t-1)$ will give a wage schedule such that $w(0) = w(1) = 0$ and $w(1/2) = 1$. One can take this exercise further by examining different wage schedules with the same present value.

3. Show anticipated versus unanticipated changes in w . An anticipated change in income at some time $s \in (0, T)$ will not affect consumption from that point in time because it has already been accounted for. An unanticipated change will permanently change consumption from the time it occurs.

4. Altruism to offspring can be analyzed as a salvage value function, taking x as an argument.

More comprehensive models as well as alternative utility functions and additional results suitable for numerical optimal control are found in Blanchard and Fisher (1989), in particular chapter 2.

SUMMARY

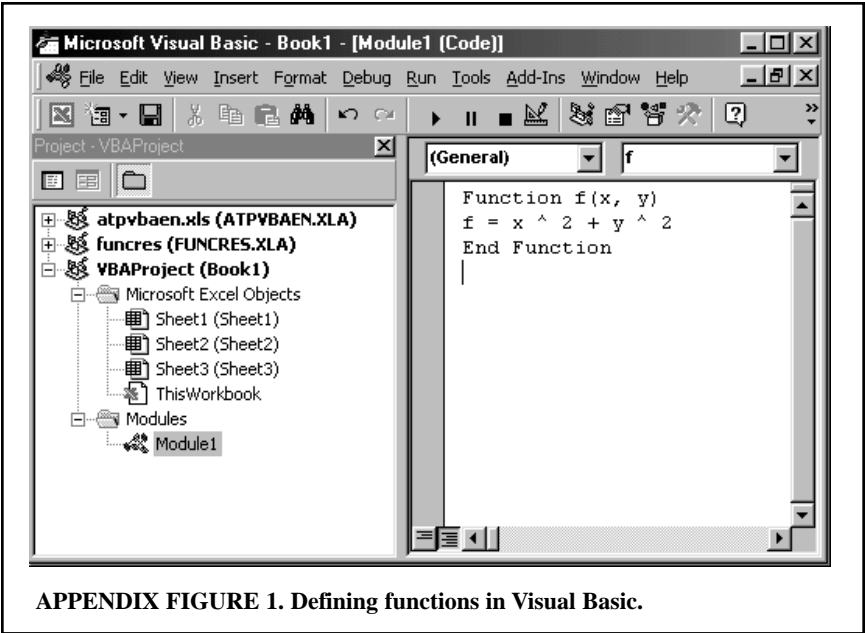
Optimal-control problems may be solved with the help of a spreadsheet. Several different problems have been discussed and solved. The method presented is probably the simplest way to numerically solve OC problems, and it has been argued that teachers and researchers may benefit from adapting the method presented here when they teach OC theory.

APPENDIX

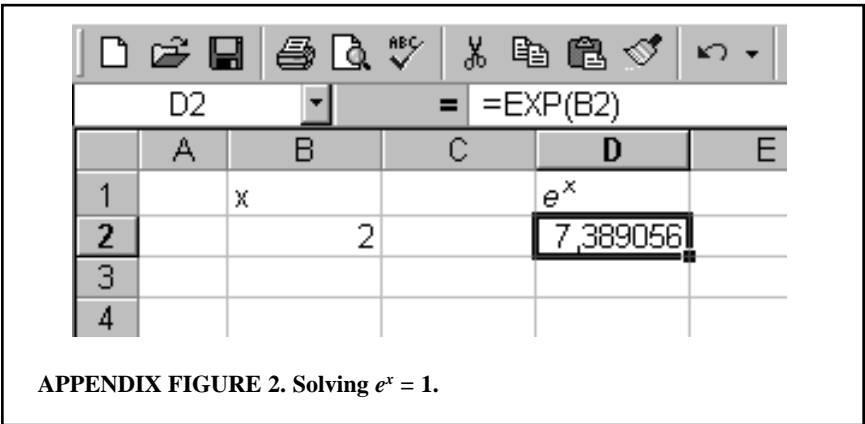
HOW TO WRITE USER-DEFINED FUNCTIONS AND SOLVE NONLINEAR EQUATIONS IN VISUAL BASIC FOR EXCEL

Writing User-Defined Functions

The main text relies heavily on the users providing the algorithm with their own user-defined functions written in Excel's version of the widely known programming language Basic. The version in Excel is specially designed to be used as a macro language, but it remains a full-featured programming language. For my purposes, I only need to know how to write functions. For example, say that I want to define a function $f(x,y) = x^2 + y^2$ and use this function to calculate $f(2,2)$

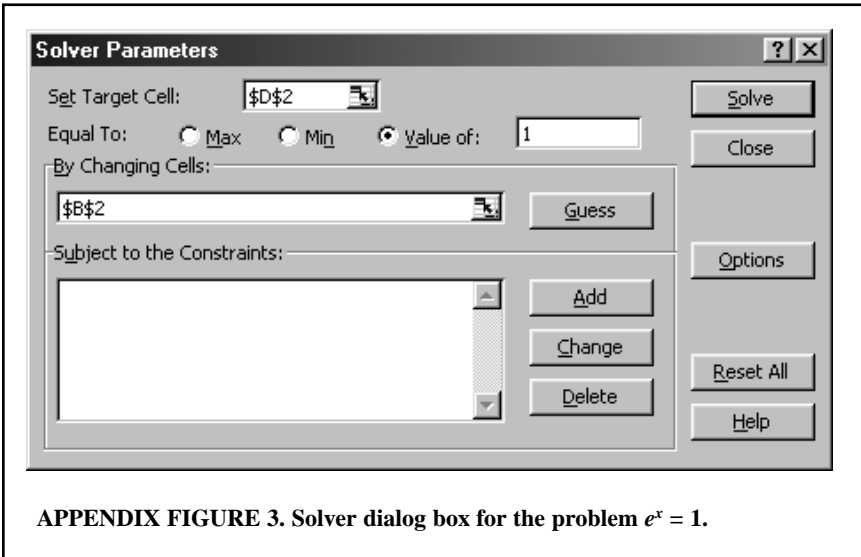


APPENDIX FIGURE 1. Defining functions in Visual Basic.



APPENDIX FIGURE 2. Solving $e^x = 1$.

$= 2^2 + 2^2$ in a spreadsheet. Start Excel, select Tools on the menu, find the item Macro, and then select Visual Basic Editor. Alternatively, after starting Excel, press <Alt> + <F11>. This will start a new program that gives access to all active Excel macros. To define a function, one must create a module. This is done by selecting the menu item Insert and then selecting the subitem Module. Take care *not* to insert a Class Module, because that will not work. This will open an area where text may be entered. After entering the function definition, the Visual Basic Editor should look something like Appendix Figure 1. After doing this, we can return to the spreadsheet by closing the Visual Basic editor. Entering $=f(2;2)$ into any cell returns the correct value 8.



APPENDIX FIGURE 3. Solver dialog box for the problem $e^x = 1$.

Solving Nonlinear Equations in Excel

When solving OC problems as specified earlier, Excel, in fact, solves a highly nonlinear set of equations. To give readers a feel for how such equations may be solved in general, I solve here a much simpler problem. Consider the equation $e^x = 1$; this equation has the solution $x = 0$. Set up the problem in a spreadsheet so that the formulas and values in the spreadsheet look like Appendix Figure 2. The value in B2 is the initial guess on x , and the value in D2 is e^2 . Then load the Solver. Simply tell the Solver that the cell D4 should be set equal to 0 by changing the cell B2. The Solver dialog box should look like Appendix Figure 3. Clicking Solve will make the Solver report the answer $x = 0$.

NOTES

1. Here I have used the option within Excel to display formulas rather than values. Check the Excel manual for details.
2. Provided that one may explicitly find u as a function of x and p .
3. Excel stores numbers as double-precision floating point. Small rounding errors often lead to answers that should be zero but are reported as close to but not exactly zero.
4. There is another way of defining constants in Excel; by defining Names in cells rather than constants in the VBA code, one could do the changes in constants in the spreadsheet rather than in the code.
5. The entire process of doing comparative dynamics can be automated and streamlined by using the Excel add-in Comparative Statics Wizard developed by Humberto Barreto and discussed in Barreto (2001).
6. It is an interesting exercise to try solving the problem with $x(T) = x_{ss}$ and comparing it to the correct procedure.

REFERENCES

Barreto, H. 2001. Teaching comparative statics with Microsoft Excel. *Journal of Economic Education* 32 (Fall): 397 (available for downloading at <http://www.wabash.edu/econexcel>).

- Blanchard, O. J., and S. Fisher. 1989. *Lectures on macroeconomics*. Cambridge, Mass.: MIT Press.
- Borrelli, R. L., and C. S. Coleman. 1987. *Differential equations: A modeling approach*. Englewood Cliffs, N.J.: Prentice-Hall.
- Clark, C. W. 1976. *Mathematical bioeconomics*. New York: John Wiley.
- Conrad, J. M. 1999. *Resource economics*. Cambridge: Cambridge University Press.
- Judd, K. L. 1999. *Numerical methods in economics*. Cambridge, Mass.: MIT Press.
- Seierstad, A., and K. Sydsæter. 1987. *Optimal control theory with economic applications*. Amsterdam: North Holland, Elsevier Science.
- Shone, R. 1997. *Economic dynamics*. Cambridge: Cambridge University Press.

We are saddened to report the death on January 30th of Kalman Goldberg, Distinguished Professor of Economics at Bradley University. Kal was the executive editor of the *Journal of Economic Education* from 1986 to 1989 and an associate editor from 1983 to 1986. Kal was a pioneer and ardent advocate of economic education for programs ranging from elementary through graduate school. Last year, Kal celebrated 50 years of teaching at Bradley University. His passing is a loss to the economic education network. He will be sorely missed.