We will not cover this lecture this year. But I make these notes available for those that might be interested in how a program such as GAMS could be used to solve a dynamic optimization problem.

I. The direct optimization approach

When you face a complicated OC problem, it is rather unlikely that you will be able to find an exact solution analytically. In practice, therefore, such problems are typically numerically. Numerical optimization programs like GAMS are designed to solve large constrained optimization problems of the form:

$$\max_z u(z) \text{ s.t. } f_i(z) \leq 0 \quad i = 1, \ldots, n$$

If the objective function and constraints are linear, then this is a linear programming problem. If the objective function is quadratic and constraints are linear, then it is a quadratic programming problem. If the objective function and constraints are nonlinear, then it is a nonlinear programming problem. GAMS (and a number of other programs) can solve any of these. However, it is worth noting that the numerical solution to an NLP problem may not be the global optimum so you should make a practice of checking your solution carefully to make sure that it is reasonable and not sensitive to starting values.

One of the most important steps that you must take to use the direct optimization approach is to discretize your problem. In the optimal control problems we’ve seen so far we have solved for $z_t$ for all $t \in [0, T]$. Clearly a computer cannot find $z$ at every point in this interval -- you must instead convert it to a discrete-time problem. This does not mean that your interval has to be 1. The interval $\Delta$ can be of any finite length you choose. The smaller $\Delta$ is, the closer your answer will be to the continuous-time problem, but the longer it will take to find the solution.

If the continuous-time problem that you are attempting to solve is

$$\max_z \int_0^T u(z_t, x_t, t) \, dt \quad \text{s.t.} \quad \dot{x} = g(z_t, x_t, t)$$

then the discrete-time analogue would be

$$\max_{\{z_t, x_t\}} \sum_{t=0}^{T/\Delta} \tilde{u}(z_t, x_t, t) \quad \text{s.t.} \quad x_{t+1} - x_t = \tilde{g}(z_t, x_t, t)$$

where $\tilde{u}(z_t, x_t, t) = \int_0^{\Delta} u(z_t, x_t, t) \, dt$ and $\tilde{g}(z_t, x_t, t) = \int_0^{\Delta} g(z_t, x_t, t) \, dt$.

Note that although $x_0$ is fixed, but $x_1, x_2, \ldots$ are somewhat flexible, although they are chosen in a sense but are bound by the state equations that govern their evolution. Hence, the $x$’s are typically represented as choice variables in the program and a dynamic optimization problem can be written as a general constrained optimization problem in which we choose the $z$’s and the $x$’s, with the exception of $x_0$. In order to solve the
problem, we also need to specify the terminal or transversality condition which would be imposed as a final constraint on the problem.
Here's the GAMS output from a slightly simplified version of the program you'll use in PS4.

**SETS**
- T TIME PERIODS /0*9/
- TFIRST(T) FIRST PERIOD
- TLAST(T) LAST PERIOD

**SCALARS**
- R DISCOUNT RATE /0.04 /
- D INTEREST RATE /0.04 /
- X0 INITIAL CAPITAL /2.00 /
- PERT A PERTURBATION FACTOR /0.05/

**PARAMETERS**
- BETA(T) DISCOUNT FACTOR

**EQUATIONS**
- TC(T) TERMINAL CONDITION
- UTIL.. UTILITY =E= SUM(T, BETA(T)*LOG(C(T)));

**MODEL**
- PS2 /XX, TC, UTIL/;

**SOLVE** PS2 MAXIMIZING UTILITY USING NLP;

**DISPLAY** C.L;

**DISPLAY** HLO.L, H.L, HHI.L;

**VARIABLES**
- X(T) - CAPITAL STOCK
- C(T) - CONSUMPTION
- UTILITY - PV OF UTILITY
- H(T) - HAMILTONIAN AT T

**PARAMETERS**
- BETA(TLAST) = BETA(TLAST)/(1-(1/(1+R)));
---- 50 SET        TFIRST        FIRST PERIOD
0

---- 50 SET        TLAST        LAST PERIOD
9

---- 61 PARAMETER BETA        DISCOUNT FACTOR
0 0.962, 1 0.925, 2 0.889, 3 0.855, 4 0.822, 5 0.790, 6 0.760
7 0.731, 8 0.703, 9 0.676

---- 98 VARIABLE C.L        - CONSUMPTION
0 0.192, 1 0.185, 2 0.178, 3 0.171, 4 0.164, 5 0.158, 6 0.152
7 0.146, 8 0.141, 9 0.135

Equation Listing    SOLVE PS2 USING NLP FROM LINE 100

---- XX        =E=  CAPITAL BALANCE
XX(1).. - 1.04*X(0) + X(1) + C(0) =E= 0 ; (LHS = -1.8877, INFES = 1.8877 ***)
XX(2).. - 1.04*X(1) + X(2) + C(1) =E= 0 ; (LHS = 0.1849, INFES = 0.1849 ***)
XX(3).. - 1.04*X(2) + X(3) + C(2) =E= 0 ; (LHS = 0.1778, INFES = 0.1778 ***)
REMAINING 6 ENTRIES SKIPPED

---- TC        =L=  TERMINAL CONDITION
TC(9).. - 1.04*X(9) + C(9) =L= 0 ; (LHS = 0.1351, INFES = 0.1351 ***)

---- UTIL        =E=  DISCOUNTED UTILITY: OBJECTIVE FUNCTION
UTIL.. - (5)*C(0) - (5)*C(1) - (5)*C(2) - (5)*C(3) - (5)*C(4) - (5)*C(5)
    - (5)*C(6) - (5)*C(7) - (5)*C(8) - (5)*C(9) + UTILITY =E= 0 ;
    (LHS = 14.7009, INFES = 14.7009 ***)

---- X        - CAPITAL STOCK
X(0)
   (.LO, .L, .UP = 2, 2, 2)
   -1.04    XX(1)
X(1)
   (.LO, .L, .UP = 0, 0, +INF)
   1       XX(1)
   -1.04    XX(2)
X(2)
   (.LO, .L, .UP = 0, 0, +INF)
   1       XX(2)
   -1.04    XX(3)
REMAINING 7 ENTRIES SKIPPED

---- C        - CONSUMPTION
C(0)
   (.LO, .L, .UP = 1.0000000E-9, 0.1923, 3)
   1       XX(1)
   (-5)    UTIL
C(1)
   (.LO, .L, .UP = 1.0000000E-9, 0.1849, 3)
   1       XX(2)
   (-5)    UTIL
C(2)
   (.LO, .L, .UP = 1.0000000E-9, 0.1778, 3)
   1       XX(3)
   (-5)    UTIL
REMAINING 7 ENTRIES SKIPPED

---- UTILITY        - PV OF UTILITY
UTILITY
   (.LO, .L, .UP = -INF, 0, +INF)
   1      UTIL
Optimal Economic Growth - PS3 - AGEC642, Spring 2000

MODEL STATISTICS

<table>
<thead>
<tr>
<th>BLOCKS OF EQUATIONS</th>
<th>3</th>
<th>SINGLE EQUATIONS</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOCKS OF VARIABLES</td>
<td>3</td>
<td>SINGLE VARIABLES</td>
<td>21</td>
</tr>
<tr>
<td>NON ZERO ELEMENTS</td>
<td>40</td>
<td>NON LINEAR N-Z</td>
<td>10</td>
</tr>
<tr>
<td>DERIVATIVE POOL</td>
<td>13</td>
<td>CONSTANT POOL</td>
<td>18</td>
</tr>
</tbody>
</table>

MODEL STATISTICS

<table>
<thead>
<tr>
<th>BLOCKS OF EQUATIONS</th>
<th>3</th>
<th>SINGLE EQUATIONS</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOCKS OF VARIABLES</td>
<td>3</td>
<td>SINGLE VARIABLES</td>
<td>21</td>
</tr>
<tr>
<td>NON ZERO ELEMENTS</td>
<td>40</td>
<td>NON LINEAR N-Z</td>
<td>10</td>
</tr>
<tr>
<td>DERIVATIVE POOL</td>
<td>13</td>
<td>CONSTANT POOL</td>
<td>18</td>
</tr>
</tbody>
</table>

RESOURCE USAGE, LIMIT

<table>
<thead>
<tr>
<th>LOWER</th>
<th>LEVEL</th>
<th>UPPER</th>
<th>MARGINAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.</td>
<td>.</td>
<td>3.899</td>
</tr>
<tr>
<td>2</td>
<td>.</td>
<td>.</td>
<td>3.749</td>
</tr>
<tr>
<td>3</td>
<td>.</td>
<td>.</td>
<td>3.605</td>
</tr>
<tr>
<td>4</td>
<td>.</td>
<td>.</td>
<td>3.467</td>
</tr>
<tr>
<td>5</td>
<td>.</td>
<td>.</td>
<td>3.333</td>
</tr>
<tr>
<td>6</td>
<td>.</td>
<td>.</td>
<td>3.205</td>
</tr>
<tr>
<td>7</td>
<td>.</td>
<td>.</td>
<td>3.082</td>
</tr>
<tr>
<td>8</td>
<td>.</td>
<td>.</td>
<td>2.963</td>
</tr>
<tr>
<td>9</td>
<td>.</td>
<td>.</td>
<td>2.849</td>
</tr>
</tbody>
</table>

EXECUTION TIME

<table>
<thead>
<tr>
<th>LOWER</th>
<th>LEVEL</th>
<th>UPPER</th>
<th>MARGINAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>-INF</td>
<td>.</td>
<td>2.740</td>
</tr>
</tbody>
</table>

EXECUTION TIME

<table>
<thead>
<tr>
<th>LOWER</th>
<th>LEVEL</th>
<th>UPPER</th>
<th>MARGINAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>-INF</td>
<td>.</td>
<td>2.740</td>
</tr>
</tbody>
</table>

---- EQU XX -- CAPITAL BALANCE

<table>
<thead>
<tr>
<th>LOWER</th>
<th>LEVEL</th>
<th>UPPER</th>
<th>MARGINAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.000</td>
<td>2.000</td>
<td>4.055</td>
</tr>
<tr>
<td>1</td>
<td>-1.833</td>
<td>+INF</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>-1.660</td>
<td>+INF</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>-1.480</td>
<td>+INF</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>-1.293</td>
<td>+INF</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>-1.098</td>
<td>+INF</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>0.895</td>
<td>+INF</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>0.684</td>
<td>+INF</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>0.465</td>
<td>+INF</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>0.237</td>
<td>+INF</td>
<td>-</td>
</tr>
</tbody>
</table>

---- EQU TC -- TERMINAL CONDITION

<table>
<thead>
<tr>
<th>LOWER</th>
<th>LEVEL</th>
<th>UPPER</th>
<th>MARGINAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>-INF</td>
<td>.</td>
<td>2.740</td>
</tr>
</tbody>
</table>

---- VAR X -- CAPITAL STOCK

<table>
<thead>
<tr>
<th>LOWER</th>
<th>LEVEL</th>
<th>UPPER</th>
<th>MARGINAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.0000E-9</td>
<td>0.247</td>
<td>3.000</td>
</tr>
<tr>
<td>1</td>
<td>1.0000E-9</td>
<td>0.247</td>
<td>3.000</td>
</tr>
<tr>
<td>2</td>
<td>1.0000E-9</td>
<td>0.247</td>
<td>3.000</td>
</tr>
<tr>
<td>3</td>
<td>1.0000E-9</td>
<td>0.247</td>
<td>3.000</td>
</tr>
<tr>
<td>4</td>
<td>1.0000E-9</td>
<td>0.247</td>
<td>3.000</td>
</tr>
<tr>
<td>5</td>
<td>1.0000E-9</td>
<td>0.247</td>
<td>3.000</td>
</tr>
<tr>
<td>6</td>
<td>1.0000E-9</td>
<td>0.247</td>
<td>3.000</td>
</tr>
<tr>
<td>7</td>
<td>1.0000E-9</td>
<td>0.247</td>
<td>3.000</td>
</tr>
<tr>
<td>8</td>
<td>1.0000E-9</td>
<td>0.247</td>
<td>3.000</td>
</tr>
<tr>
<td>9</td>
<td>1.0000E-9</td>
<td>0.247</td>
<td>3.000</td>
</tr>
</tbody>
</table>

---- VAR C -- CONSUMPTION

<table>
<thead>
<tr>
<th>LOWER</th>
<th>LEVEL</th>
<th>UPPER</th>
<th>MARGINAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.0000E-9</td>
<td>0.247</td>
<td>3.000</td>
</tr>
<tr>
<td>1</td>
<td>1.0000E-9</td>
<td>0.247</td>
<td>3.000</td>
</tr>
<tr>
<td>2</td>
<td>1.0000E-9</td>
<td>0.247</td>
<td>3.000</td>
</tr>
<tr>
<td>3</td>
<td>1.0000E-9</td>
<td>0.247</td>
<td>3.000</td>
</tr>
<tr>
<td>4</td>
<td>1.0000E-9</td>
<td>0.247</td>
<td>3.000</td>
</tr>
<tr>
<td>5</td>
<td>1.0000E-9</td>
<td>0.247</td>
<td>3.000</td>
</tr>
<tr>
<td>6</td>
<td>1.0000E-9</td>
<td>0.247</td>
<td>3.000</td>
</tr>
<tr>
<td>7</td>
<td>1.0000E-9</td>
<td>0.247</td>
<td>3.000</td>
</tr>
<tr>
<td>8</td>
<td>1.0000E-9</td>
<td>0.247</td>
<td>3.000</td>
</tr>
<tr>
<td>9</td>
<td>1.0000E-9</td>
<td>0.247</td>
<td>3.000</td>
</tr>
</tbody>
</table>

---- VAR UTILITY -- PV OF UTILITY

<table>
<thead>
<tr>
<th>LOWER</th>
<th>LEVEL</th>
<th>UPPER</th>
<th>MARGINAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-INF</td>
<td>-11.356</td>
<td>+INF</td>
</tr>
</tbody>
</table>

---- REPORT SUMMARY : 0 NONOPT 0 INFEASIBLE 0 UNBOUNDED 0 ERRORS
Optimal Economic Growth - PS3 - AGEC642, Spring 2000

---- 116 VARIABLE HLO.L - HAMILTONIAN AT T WITH SMALL PERTURBATION DOWN
0  -199.704,  1  -194.521,  2  -189.538,  3  -184.746,  4  -180.139
5  -175.709,  6  -171.449,  7  -167.354,  8  -163.415,  9  -98.048

---- 116 VARIABLE H.L - HAMILTONIAN AT T
0  -199.579,  1  -194.402,  2  -189.423,  3  -184.636,  4  -180.033
5  -175.703,  6  -171.443,  7  -167.348,  8  -163.324,  9  -94.583

---- 116 VARIABLE HH.L - HAMILTONIAN AT T WITH SMALL PERTURBATION UP
0  -199.696,  1  -194.513,  2  -189.531,  3  -184.739,  4  -180.132
5  -175.703,  6  -171.444,  7  -167.348,  8  -163.409,  9  -91.287

EXECUTION TIME = 0.000 SECONDS 0.9 Mb WIN-18-095

USER: Agricultural Economics
Texas Agricultural Exp. Sta.

**** FILE SUMMARY

INPUT  D:\INSTRUCTION\642DYNAMICS\PS'S\RAMSEY.GMS
OUTPUT D:\INSTRUCTION\642DYNAMICS\PS'S\RAMSEY.LST
II. An application of numerical optimal control - Chatterjee, Howitt and Sexton (1998)

This paper provides a nice example of applied use of dynamic optimization. The authors do a good job of setting up and motivating the problem, making the conceptual links to dynamic optimization and then clearly present results of a numerical exercise. We will discuss the paper referring to a handout with the most important that will be provided in class.

III. The backward sweep algorithm (optional)

The backward sweep algorithm (BSA) is a relatively straightforward but not terribly efficient way to solve simple one-dimensional optimal control problems. The algorithm does, however, have three pedagogical advantages: 1) it uses the maximum conditions of optimal control and it is relatively easy to program; 2) it's a good way to get started programming in Visual Basic; and 3) it demonstrates the numerical technique of searching for the solution to the problem by gradually improving our guesses.

Suppose we're interested in solving the problem:

\[
\max_z \int_0^T e^{-nt} u(z_t, x_t) dt \quad \text{s.t.} \quad \dot{x}_t = f(z_t, x_t)
\]

with \(x_T\) free so that our transversality condition is \(\lambda_T = 0\). Furthermore, suppose that for some reason it is difficult or impossible to find a closed form solution to the problem, perhaps because \(f\) is messy or the resulting differential equations cannot be solved. So we decide to use numerical methods to solve the problem instead.

To use the backward sweep algorithm you need to discretize the problem. So instead of an integral we use the discounted sum of utility from 0 to \(T\) and we have to change \(\dot{x}\) equation to \(x_{t+1} = x_t\). Choosing the appropriate time step is important. Sometimes we could think of this as years, but other times it might be days.

Your solution will be three vectors with \(z, x\) and \(\lambda\), all from zero to \(T\). You would also want to present the discounted value of the present value of your sum.

The BSA makes extensive use of the maximum conditions of the Hamiltonian,

A) \(\frac{\partial H}{\partial z} = 0\) and

B) \(\frac{\partial H}{\partial x_t} = -(\lambda_{t+1} - \lambda_t)\)

The basic steps of the BSA are as follows:
1. Initialize the arrays for \(z, x\) and \(\lambda\).
2. Make a feasible guess as to the value for \(z\) over the time horizon. The better your guess, the faster you'll get convergence and the better will be your results.
3. Set $x(0) = x_0$.
4. Calculate the vector $x_1, x_2, \ldots, x_T$ associated with your initial guess at $z$ using the state equation.
5. Use your transversality condition to calculate $\lambda_T$. If $x_T$ is free then this is easily found by setting $\lambda_T = 0$. If $x_T$ is not free (as in PS#3) you'll need to find the $\lambda_T$ that would push $x_{T-1}$ to zero.
6. Using the adjoint equation, B, work backwards finding $\lambda_{T-1}, \lambda_{T-2}, \ldots, \lambda_0$.

**Iteratively find the optimal values of $z$, $x$ and $\lambda$.**

7. In finding our next estimate of $z$ we will use solution to A. If $z(x, \lambda)$ is the value of $z$ that solves $A$, then our new guess of $z_0$ would be a partial step toward $z(x, \lambda)$, i.e., $z_0^2 = \theta \cdot z(x_0, \lambda_0) + (1 - \theta)z_0^1$ where $z_0^2$ is the second estimate of $z_0$ and $z_0^1$ is the first estimate. The choice of the parameter $\theta$ is very critical! If it is too large the algorithm will blow up.

8. Calculate $x_1$ using $x_{t+1} = x_t + f(z_t, x_t)$.
9. Repeat 7 and 8 until the full vector $z$ and $x$ are estimated.
10. Now, we sweep backwards calculating $\lambda$ by using B just as in 5 and 6.
11. Repeat 10 for $T-2, T-3,$ etc. until you get back to $t=0$.

**Check for convergence**

12. Compare your new estimate of $z$ with the previous estimate of $z$. If they are very close to each other, then we say that the algorithm has converged and we jump to 14. If they are not close enough, go through the process again starting at 7. (see note on convergence below).

**Scroll up and down again and again until convergence is met or until the maximum number of iterations is reached**

13. Go back to 7 and repeat until a convergence criterion is met. **You're done**

14. Save your results to a file. If convergence criterion has not been met, look at your output and see what's going wrong.

**A word on convergence of numerical algorithms**

Generally, the convergence criterion is some measure of the "distance" between the $k^{th}$ and the $k+1^{th}$ iteration. For example, you may want to use $\sum_{t=0}^{T} \left( z_t^k - z_t^{k-1} \right)^2 \leq \varepsilon$ or $\max_t |z_t^k - z_t^{k-1}| \leq \varepsilon$. In the program below I use the , two options are available, diff1 = the difference between the z-vectors, and diff2 = the distance in the value function from one iteration to the next.
Option Explicit
Dim z() As Double, zstore() As Double, L() As Double, x() As Double
Dim Val As Double, ValOld As Double, Theta As Double, diff As Double
Dim d As Double, x0 As Double, r As Double, eps As Double, DiffOld
Dim it As Integer, iIter As Integer, maxiter As Integer, T As Integer
' I used the watch variables for debugging. This allows me to look at one value rather than the full array.
Dim zwatch, xwatch, xnextwatch, lwatch
Sub BackSweep()
' ----------------------------
' Initialize values
' ----------------------------
T = 19
x0 = 3#
r = 0.03
d = 0.07
eps = 0.00000001
Theta = 0.005
maxiter = 500
' ----------------------------
' Redimension the arrays
' ----------------------------
ReDim z(T)
ReDim x(T + 1)
ReDim zstore(T)
ReDim L(T)
ReDim zstore(T)
' --------------------
' Name the ranges where we'll be writing output
' (note that ranges start at row 3 so that I can use row 2 for the O'th entry)
' --------------------
Range("a1") = "t"
Range("a3:a100").Name = "t"
Range("b1") = "X"
Range("b3:b100").Name = "X"
Range("c1") = "Z"
Range("c3:c100").Name = "Z"
Range("d1") = "Lambda"
Range("d3:d100").Name = "Lambda"
' 'Set the initial values of x and z by looping from iT=0 to iT=T,
' arbitrarily picking the values of z, then calculating x
' Write these values to the spreadsheet.
' ' For it = 0 To T
' xwatch = x(it)
' z(it) = x(it) / (T - it + 1)
' xnextwatch = x(it + 1)
' Range("T").Cells(it) = it
' Range("x").Cells(it) = x(it)
' Range("z").Cells(it) = z(it)
' Next it
' ' Initialize values of L. L(T) uses the transversality condition.
' Then we calculate L(t-1) using the maximum condition,
' dl/dx=[-(L(t+1)-L(t))]
' ' L(T) = Exp(-r * T) / (x(T) * (1 + d))
' Range("Lambda").Cells(T) = L(T)
' For it = 1 To T
' L(T - it) = L(T - it + 1) / (1 - d)
' Range("Lambda").Cells(T - it) = L(T - it)
' Next it
' ' Enter Main portion of program
' Repeatedly sweep forward, taking a partial step from your current x to z that follows from the FOC. Then sweep backward as we did below.
' For iIter = 1 To maxiter
' Range("f1") = iIter
' Call ForwardSweep
' Call BackwardSweep
' Call Convergence
' Range("e1") = "The algorithm converged in"
' Range("e2") = iIter
' Range("e3") = "iterations."
' If diff < eps Then Exit For
' End If
' Next iIter
' Once Convergence has been reached, write your results to the spreadsheet
' For it = 0 To T
' Range("x").Cells(it) = x(it)
' Range("z").Cells(it) = z(it)
' Range("x").Cells(T + 1) = x(T + 1)
' Range("Lambda").Cells(it) = L(it)
' Next it
End Sub
Sub ForwardSweep()
' Forward sweep
' H = exp(−r*t)∗ln(c(t)) +L(t)*(x(t) − c(t))
' FOC: exp(−r*t)/z(t) = L(t)
' z(t) = exp(−r*t)/L(t)
' State equation
' x(t+1) = x(t)∗(1+d) − c(t)
' note that x(0) is not changed
' The if statement ensures that x never goes below zero
' ' For it = 0 To T
' xwatch = x(it)
' z(it) = Theta * (Exp(-r * it) / L(it)) + (1 - Theta) * z(it)
' xnextwatch = x(it + 1)
' Range("T").Cells(it) = it
' Range("x").Cells(it) = x(it)
' Range("z").Cells(it) = z(it)
' Next it
' Use FOC to find z*
' Check for feasibility
' If z(it) > x(it) * (1 + d) Then
' z(it) = (x(it) * (1 + d))/2
' End Sub
Sub BackwardSweep()
' Backward sweep
' H = exp(−r*T)∗ln(c(T)) +L(T)*(x(T) − c(T))
' FOC: exp(−r*T)/z(T) = L(T)
' z(T) = exp(−r*T)/L(T)
' State equation
' x(T) = x(T)∗(1+d) − c(T)
' note that x(0) is not changed
' The if statement ensures that x never goes below zero
' ' For it = T To 0
' xnextwatch = x(it)
' z(it) = Theta * (Exp(-r * T) / L(T)) + (1 - Theta) * z(it)
' xwatch = x(it)
' Range("T").Cells(it) = it
' Range("x").Cells(it) = x(it)
' Range("z").Cells(it) = z(it)
' Next it
'
zwatch = z(it)
End If
'
State equation
x(it + 1) = x(it) * (1 + d) - z(it)
xnextwatch = x(it + 1)
Next it
End Sub
Sub BackwardSweep()
L(T) = 0
L(T) = Exp(-r * T) / (x(T) * (1 + d))
For it = 1 To T
L(T - it) = L(T - it + 1) / (1 - d)
lwatch = L(T - it)
Next it
End Sub
Sub Convergence()
ValOld = Val
DiffOld = diff
diff = 0#
Val = 0#
For it = 0 To T
Val = Val + Log(z(it)) / ((1 + r) ^ it)
Next it
diff = Abs(Val - ValOld)
' Adjust theta down if steps are too big
If diff > DiffOld Then
Theta = Theta / 2
Else
' Increase theta if steps are proceeding o.k.
Theta = Theta * 1.1
End If
End Sub