COMPUTATIONAL ECONOMICS

Thrift in GAMS

David A. Kendrick
Ruben P. Mercado
Hans M. Amman

1 Introduction

Many students face a tough financial problem—their expenses exceed their income. Thus they must work to supplement their income and/or borrow money from student loan funds. This familiar student situation provides a good setting to learn about dynamic personal financial planning models financial planning models.

Some students have financial assets such as stocks and bonds; however, many others have few assets and substantial liabilities in the form of credit card debt and student loans. In this chapter we provide a model in which a student can hold assets in either low-interest bonds or higher-interest stocks. The student can also hold some assets in a checking account while paying living expenses out of that account and depositing earnings from part-time work into the account. If living expenses exceed earnings then the student must either draw down stock and bond assets or else borrow from a student loan fund at a low interest rate or from a credit card firm at a much higher interest rate.

We begin with a simple version of the model with only bonds, a checking account, and student loans and advance to a more complex model later in the chapter.

2 The Mathematics of the Thrift Model

Consider a student who has a checking account as well as some money saved in government bonds. The dynamic equation for the bonds held by the student can be written as

$$Sb_{t+1} = (1 + rb)Sb_t - Xbc_t + Xcb_t \tag{1}$$

where

Sb = stock of bonds
rb = rate of interest on bonds
Xbc = transfer from bonds to checking account
Xcb = transfer from checking account to bonds

Thus the stock of bonds next period is equal to the stock of bonds this period multiplied by one plus the interest rate on the bonds minus bonds that are cashed in (Xbc) plus new bonds that are purchased (Xcb). As is shown below in equation (2), the proceeds from the sales of bonds (Xbc) are deposited in the student's checking account. Thus the equation for his or her checking account can be written as

$$Sc_{t+1} = (1 + rc)Sc_t + Xbc_t - Xcb_t \tag{2}$$

where

$$Sc = \text{stock of funds in the checking account}$$
$$rc = \text{rate of interest on funds in the checking account}$$

Likewise additional bonds can be purchased by withdrawing money from the checking account, Xcb. The bond and checking accounts are both asset accounts. One can also create a liability account in the form of a student loan equation that is the amount the student owes to the bank:

$$Ssl_{t+1} = (1 + rsl)Ssl_t - Xcsl_t + Xslc_t \tag{3}$$

where

$$\begin{aligned} Ssl &= \text{stock of student loans} \\ rsl &= \text{rate of interest on student loans} \\ Xcsl &= \text{transfer from checking account to student loans} \\ Xslc &= \text{transfer from student loans to checking account} \end{aligned}$$

In this equation $Xcsl$ is the amount that the student withdraws from the checking account to repay student loans and $Xslc$ is the amount borrowed from the loan account to deposit in the checking account. Given these additional flows to and from the checking account we need to modify the checking account state equation from equation (2) by including two more terms so that it becomes

$$Sc_{t+1} = (1 + rc)Sc_t + Xbc_t - Xcb_t - Xcsl_t + Xslc_t \tag{4}$$

In addition, the student has a part-time job and deposits these wages, Wa, into the checking account and pays his or her living expenses, Le, from the account, so we have to include two more terms; thus equation (4) becomes

$$Sc_{t+1} = (1 + rc)Sc_t + Xbc_t - Xcb_t - Xcsl_t + Xslc_t + Wa_t - Le_t \tag{5}$$

The model then consists of the bond equation (1), the checking account equation (2), and the student loan equation (3) and can be written in matrix form as

$$\begin{aligned} \begin{bmatrix} Sb \\ Sc \\ Ssl \end{bmatrix}_{t+1} &= \begin{bmatrix} 1+rb & 0 & 0 \\ 0 & 1+rc & 0 \\ 0 & 0 & 1+rsl \end{bmatrix} \begin{bmatrix} Sb \\ Sc \\ Ssl \end{bmatrix}_t + \\ & \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & -1 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} Xbc \\ Xcb \\ Xcsl \\ Xslc \end{bmatrix}_t + \begin{bmatrix} 0 & 0 \\ 1 & -1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} Wa \\ Le \end{bmatrix}_t \end{aligned} \tag{6}$$

or in vector difference equation form as

$$x_{t+1} = Ax_t + Bu_t + Cz_t \tag{7}$$

where the state vector x_t, the control vector u_t, and the exogenous vector z_t are defined as

$$x_t = \begin{bmatrix} Sb \\ Sc \\ Ssl \end{bmatrix}_t \quad u_t = \begin{bmatrix} Xbc \\ Xcb \\ Xcsl \\ Xslc \end{bmatrix}_t \quad z_t = \begin{bmatrix} Wa \\ Le \end{bmatrix}_t \tag{8}$$

and the matrices A, B, and C are

$$A = \begin{bmatrix} 1+rb & 0 & 0 \\ 0 & 1+rc & 0 \\ 0 & 0 & 1+rsl \end{bmatrix} B = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & -1 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix} C = \begin{bmatrix} 0 & 0 \\ 1 & -1 \\ 0 & 0 \end{bmatrix} \tag{9}$$

Difference equations models like equation (7) are frequently called *system* equations and are widely used in engineering and in economics to represent dynamic systems. Such models also often have a criterion function that is optimized subject to the system equations.

A common form of the criterion function is the quadratic tracking function, which is different than the usual utility maximization criterion used in consumer theory, the cost minimization criterion sometimes used in production theory, or the terminal wealth maximization sometimes used in portfolio models. The quadratic tracking criterion function includes desired paths for the state variables and the control variables and seeks to minimize the weighted squared separation between the desired and the optimal paths. For example, an individual may wish to save over the course of a lifetime for multiple purposes, such as purchasing a house or car, paying for college for children, and providing retirement income. The individual may also want to keep a target amount in a checking account. So the desired time path for savings accounts, stock and bond holdings, and checking account balances may be time varying and have a complicated shape. Moreover, some goals may be more important than others and thus have higher weights attached to them.

A static version of the quadratic tracking criterion function with only a single state variable x and a single control variable u can be written:

$$J = w\,(x - \tilde{x})^2 + \lambda\,(u - \tilde{u})^2 \tag{10}$$

where

$J =$ criterion value
$\tilde{x} =$ desired value of the state variable
$\tilde{u} =$ desired value of the control variable
$w =$ priority on the state variable
$\lambda =$ priority on the control variable

Since w and λ are positive and the goal is to minimize J, one wants the state variable x to be as close as possible to its desired value $\tilde{x}$ and the control variable u as close as possible to its desired value $\tilde{u}$. Thus it is obvious that the criterion function in this case can be minimized by setting

$$x = \tilde{x} \qquad u = \tilde{u} \tag{11}$$

However, this is usually not possible because the state and control variables are related to one another through the system equation, so there is usually a trade-off between having x as close as possible to $\tilde{x}$ and u as close as possible to $\tilde{u}$. Furthermore, this trade-off is affected by the priority parameters w and λ. Thus if w is large and λ is small the optimal solution is to set x close to $\tilde{x}$ and u not so close to $\tilde{u}$.

The priority parameters w and λ are also sometimes called penalty weights, depending on whether one is thinking of them positively as priorities or negatively as penalties in a criterion function that is to be minimized. Both terms are used in this book and elsewhere in the literature.

When the state variable and the control variable are not scalars but rather vectors, equation (10) can be written in vector-matrix form as

$$J = (x - \tilde{x})'\,W\,(x - \tilde{x}) + (u - \tilde{u})'\,\Lambda\,(u - \tilde{u}) \tag{12}$$

where

$x =$ state vector
$u =$ control vector
$\tilde{x} =$ desired value of the state vector
$\tilde{u} =$ desired value of the control vector
$W =$ diagonal priority matrix for the state vector
$\Lambda =$ diagonal priority matrix for the control vector

Consider only the first term on the right-hand side of equation (12). For the case at hand it can be written as

$$(x-\tilde{x})' W (x-\tilde{x}) = \begin{pmatrix} Sb - S\tilde{b} \\ Sc - S\tilde{c} \\ Ssl - Ss\tilde{l} \end{pmatrix}' \begin{bmatrix} wb & 0 & 0 \\ 0 & wc & 0 \\ 0 & 0 & wsl \end{bmatrix} \begin{pmatrix} Sb - S\tilde{b} \\ Sc - S\tilde{c} \\ Ssl - Ss\tilde{l} \end{pmatrix} \tag{13}$$

where

$$\begin{aligned} wb &= \text{priority for bonds} \\ wc &= \text{priority for checking account} \\ wsl &= \text{priority for student loan account} \end{aligned}$$

Taking the transpose of the first vector on the right-hand side of equation (13) and doing the matrix vector multiplication of the remaining matrix and vector in that equation yields

$$(x-\tilde{x})' W (x-\tilde{x}) = \begin{pmatrix} Sb - S\tilde{b} & Sc - S\tilde{c} & Ssl - Ss\tilde{l} \end{pmatrix} \begin{bmatrix} wb\left(Sb - S\tilde{b}\right) \\ wc\left(Sc - S\tilde{c}\right) \\ wsl\left(Ssl - Ss\tilde{l}\right) \end{bmatrix} \tag{14}$$

or

$$(x-\tilde{x})' W (x-\tilde{x}) = wb\left(Sb - S\tilde{b}\right)^2 + wc\left(Sc - S\tilde{c}\right)^2 + wsl\left(Ssl - Ss\tilde{l}\right)^2 \tag{15}$$

Since the W matrix is diagonal the quadratic form on the left-hand side of equation (15) is equal to a weighted sum of squares of the difference between each state variable and its desired value, with the weights being the respective priorities.

From a similar set of mathematical statements we can show that the quadratic form in the control variables in equation (12) is

$$\begin{aligned} (u-\tilde{u})' \Lambda (u-\tilde{u}) = \lambda bc\left(Xbc - Xb\tilde{c}\right)^2 + \lambda cb\left(Xcb - Xc\tilde{b}\right)^2 \\ +\lambda csl\left(Xcsl - Xcs\tilde{l}\right)^2 + \lambda slc\left(Xslc - Xsl\tilde{c}\right)^2 \end{aligned} \tag{16}$$

where

$$\begin{aligned} \lambda bc &= \text{priority on transfers from bonds to checking} \\ \lambda cb &= \text{priority on transfers from checking to bonds} \\ \lambda csl &= \text{priority on transfers from checking to student loan} \\ \lambda slc &= \text{priority on transfers from student loan to checking} \end{aligned}$$

The priorities on the control variables in the λ parameters work analogously to those on the state variables, that is, a large value for the priority indicates that the student wants to hold that control variable close to its desired values. Of course, what really matters is not the absolute values of the w and λ priorities but their values *relative* to one another. So the student who wants to ensure that the state variables reach their desired values assigns relatively high priorities to the state variables with the w parameters and relatively low priorities to the control variables with the λ parameters.

In summary, we can write the quadratic tracking criterion function for a single period as

$$J = (x - \tilde{x})' W (x - \tilde{x}) + (u - \tilde{u})' \Lambda (u - \tilde{u}) \tag{17}$$

However, we want to use this criterion function in a multiperiod model, so we need a dynamic version of equation (17), which can be written

$$\begin{aligned} J = \; & \tfrac{1}{2} \; (x_N - \tilde{x}_N)' W_N (x_N - \tilde{x}_N) + \\ & \tfrac{1}{2} \sum_{t=0}^{N-1} \left[(x_t - \tilde{x}_t)' W (x_t - \tilde{x}_t) + (u_t - \tilde{u}_t)' \Lambda (u_t - \tilde{u}_t) \right] \end{aligned} \tag{18}$$

It is customary to include the 1/2 fractions in the criterion function so that when the derivatives of this quadratic function are taken the first-order conditions do not include a 2. A distinction is also drawn between the priorities on the state variables in the terminal time period N, that is, W_N, and those in all other time periods, W. This permits different priorities to be attached to the state variables in the terminal period than in other periods. Moreover, the control vector for the terminal period $u_N$does not appear in the criterion since it does not affect the state until period $N + 1$, and that period is not included in the model.

In summary the dynamic control theory model seeks to find the control variables

$$(u_0, u_1, \cdots, u_{N-1})$$

that minimize the criterion function

$$\begin{aligned} J = \; & \tfrac{1}{2} \; (x_N - \tilde{x}_N)' W_N (x_N - \tilde{x}_N) + \\ & \tfrac{1}{2} \sum_{t=0}^{N-1} \left[(x_t - \tilde{x}_t)' W (x_t - \tilde{x}_t) + (u_t - \tilde{u}_t)' \Lambda (u_t - \tilde{u}_t) \right] \end{aligned} \tag{19}$$

subject to the systems equations [from equation (7)]

$$x_{t+1} = Ax_t + Bu_t + Cz_t \tag{20}$$

with the initial conditions

$$x_0 \text{ given} \tag{21}$$

The student financial model described above can be specified in this form using the state, control, and exogenous variable vectors:

$$x_t = \begin{bmatrix} Sb \\ Sc \\ Ssl \end{bmatrix}_t \quad u_t = \begin{bmatrix} Xbc \\ Xcb \\ Xcsl \\ Xslc \end{bmatrix}_t \quad z_t = \begin{bmatrix} Wa \\ Le \end{bmatrix}_t \tag{22}$$

and the matrices A, B, and C

$$A = \begin{bmatrix} 1+rb & 0 & 0 \\ 0 & 1+rc & 0 \\ 0 & 0 & 1+rsl \end{bmatrix} B = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & -1 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix} C = \begin{bmatrix} 0 & 0 \\ 1 & -1 \\ 0 & 0 \end{bmatrix} \tag{23}$$

with the exogenous variables $(z_0, z_1, \cdots, z_{N-1})$ and the initial state x_0 given.

sectionThe Evanchik Model

A model of this form was developed by Michael Evanchik (1998) when he was an undergraduate in a Computational Economics class at the University of Texas. Evanchik's model is slightly more complicated than the model described earlier in that it includes three types of assets rather than two by adding equities to the bond and checking accounts. Moreover, the model includes two liabilities rather than one since the student can borrow not only from a student loan account but also from that most popular source of student support—a credit card account. Thus the state vector for this model has five variables, that is,

$$x_t = \begin{bmatrix} Sb \\ Se \\ Sc \\ Scc \\ Ssl \end{bmatrix}_t \tag{24}$$

where

$Sb =$ stock of bonds
$Se =$ stock of equities
$Sc =$ stock of funds in the checking account
$Scc =$ stock of credit card loans
$Ssl =$ stock of student loans

Correspondingly, the control vector ut includes more elements to permit a variety of transfers among these accounts:

$$u_t = \begin{bmatrix} Xbe \\ Xbc \\ Xbcc \\ Xbsl \\ Xec \\ Xecc \\ Xesl \\ Xcacc \\ Xcsl \\ Xccsl \end{bmatrix} \tag{25}$$

where

$Xbe =$ transfer from bonds to equities
$Xbc =$ transfer from bonds to checking account
$Xbcc =$ transfer from bonds to credit card account
$Xbsl =$ transfer from bonds to student loans
$Xec =$ transfer from equities to checking account
$Xecc =$ transfer from equities to credit card account
$Xesl =$ transfer from equities to student loans
$Xcacc =$ transfer from checking account to credit card account
$Xcsl =$ transfer from checking account to student loan account
$Xccsl =$ transfer from credit card account to student loan account

There is one anomaly in the foregoing variable naming scheme, which has been introduced to eliminate a source of confusion. The transfer that would have been labeled $Xccc$ to be consistent with all of the rest of the variable names has been labeled $Xcacc$ instead to make it clear that the transfer is from the checking account to the credit card account rather than vice versa.

A GAMS version of the model was created by one of the authors of this chapter, Genevieve Solomon, while she was a student in the same class a couple of years after Evanchik. She added a third exogenous variable to the two already in the

Evanchik model so that the exogenous variable vector for the model in this chapter is

$$z_t = \begin{bmatrix} Wa \\ Le \\ Sh \end{bmatrix}_t \tag{26}$$

where

$$\begin{aligned} Wa &= \text{wages} \\ Le &= \text{living expenses} \\ Sh &= \text{scholarship} \end{aligned}$$

The GAMS version of the thrift model has the useful property that it is possible to put explicit upper bounds on the state and control variables. For example, there are frequently upper bounds on how much money a student can borrow per semester from the student loan organization, and credit cards also frequently have upper bounds on the amount that a student can borrow.

3 The Model in GAMS

The GAMS program corresponding to the thrift problem is available at the book web site. The first step for the GAMS version of the model is to define the sets, of which there are four: state variables, control variables, exogenous variables, and the time horizon. These sets are declared and defined in GAMS as follows: DKDavid Kendrick-1499962496Same problem – same solution in the following 5 lines of code. It is necessary to reduce the font size from Courier 12 to Courier 10 to avoid a spurious carriage return in the line that begins `t horizon`

```
Sets n  states     / Sb, Se, Sc, Scc, Ssl /

     m  controls   /Xbe, Xbc, Xbcc, Xbsl, Xec,

     Xecc, Xesl, Xcacc, Xcsl, Xccsl/

     k  exogenous  / Wa, Le, Sh /

     t  horizon    / 2000, 2001, 2002, 2003, 2004 /
```

Note here how sets are specified in the GAMS language. In this model there are five GAMS language. In this model there are five state variables. The set of states could be specified in mathematics as

$$N = \{Sb,\ Se,\ Sc,\ Scc,\ Ssl\}$$

and the equivalent GAMS statement would be

```
n = / Sb, Se, Sc, Scc, Ssl /
```

GAMS has forward slashes as set delimiters, whereas mathematics has braces. This means that you should be very careful not to use forward slashes in a GAMS model in text statements like dollars/ton since the slash confuses the GAMS compiler and may result in an error. In addition, we include the word *state* in the statement for the set of state variables, which is the text associated with the set n. Thus the complete statement in GAMS for the set n is

```
n  states     / Sb, Se, Sc, Scc, Ssl /
```

Also, similar statements are used to declare and define the sets of control variables m, exogenous variables k, and time periods t. Next three subsets of the time set are declared. In many computer languages there is a distinction between *declaring* an element and *defining* it. That distinction is also used here since the statements that follow are used to declare three sets that are defined later when the elements in the set are determined:

```
tu(t) control horizon

ti(t) initial period

tz(t) terminal period ;
```

The control horizon, initial period, and the terminal period sets are important later for the equations. Further, in these three statements the (t) is used to indicate that the preceding set, namely, `tu`, is a subset of the set t.

Here in the body of this chapter we introduce the parts of the GAMS statement of the model one section at a time. Later you may want to look at the entire GAMS statement of the model, shown in Appendix A at the end of the chapter. Next tables are created to represent the matrices in the systems equations ((20):

```
A   (one plus interest rates) (5 $\times$ 5)

B   (direction of transfers) (5 $\times$ 10)

C   (exogenous variable signs) (5 $\times$ 3)
```

```
w   (state variable penalty matrix) (5 $\times$ 5)

wn (state variable penalty matrix for the terminal period) (5 $\times$ 5)

lambda (control variable penalty matrix) (10 $\times$ 10)
```

In doing this we also need an alias statement,

```
Alias (n,np), (m,mp) ;
```

which simply makes a copy of the set n and calls it np (n prime) and of the set m and calls it mp (m prime). This alias is necessary for setting up summations for matrix operations in GAMS. Also the aliases are needed when matrices are transposed in later equations. The next part of the input defines the time subsets, tu, ti, and tz of the full time set t. The first of these, tu, is the set of all time periods other than the terminal period and is defined with the GAMS statement

```
tu(t) = yes\$(ord(t) lt card(t));
```

This statement makes use of two GAMS keywords ord and card, which are operators defined on sets; card is an abbreviation for cardinal, which is the number of elements in the set. Consider a set S in mathematics:

$$S = \{a, b, c, d\}$$

The cardinality of this set is four since it has four elements. In contrast ord is an abbreviation for ordinal, which represents the ordinal position of each element in the set. Thus the element c is in the third ordinal position in the above set. So the GAMS statement defining the set tu can be read as, *`tu` is the set of elements whose ordinal position in the set t are strictly less than the cardinality of the set.* Thus, recalling that the set t is

```
tu =    / 2000, 2001, 2002, 2003, 2004 /
```

we see that the set tu is

```
tu =    / 2000, 2001, 2002, 2003 /
```

that is, it is all the elements in the set t except for the last one. The second of the subsets of t is the set ti, which is the initial time period only. It is defined with the GAMS statement

```
ti(t) = yes\$(ord(t) eq 1);
```

Thus the set ti contains the element that is in the first position in the set t, namely 2000. The third subset of t is defined with the GAMS statement

```
 tz(t) = not tu(t);
```

Thus tz is the set of all elements in the set t that are not in the set tu, and that is only the last element, namely 2004. Finally, just as a check, the elements of the full set and the three subsets are displayed in the output file with the statement

```
 Display t, ti, tz, tu;
```

When doing set manipulations in GAMS it is useful to display the results as a check against errors. Once the sets are specified, then the data can be input using the table and parameter keywords as shown in what follows,. Consider first the use of the table keyword to input the A matrix:

```
Table a(n,np)  state vector matrix

        Sb     Se     Sc      Scc     Ssl

Sb     1.05

Se            1.10

Sc                   1.01

Scc                           1.13

Ssl                                   1.03
```

Observe that the parameter a is followed by the sets over which it is defined, that is, it is written as a(n,np). It is not necessary to include the sets here; however, it is a useful precaution because when the sets are provided the GAMS complier can check to ensure that all the element names used in the input of the table do indeed belong to the appropriate sets for the rows and columns of the table. Thus if the user misspells an element in the table input GAMS issues a warning. Following the name of the table and its set is a line of text, that is,

```
       state vector matrix
```

The ability to use text like this phrase makes GAMS statements much easier to read and understand. The convention in GAMS is that the absence of an explicit data entry in a table results in that element of the matrix being set to zero. So all the elements in the A matrix others than those on the diagonal are set equal to zero.

Recall that the diagonal elements in the A matrix are 1 plus the appropriate interest rate. So the interest rate is 5 percent on bonds, 10 percent on equities, and only 1 percent on checking accounts. (Of course bonds and equities carry greater risk than checking accounts. Comparative risk is not addressed in this model but is included in the models on portfolio selection used later in this book.) One way to alter the model to better represent the financial condition of a given individual is to change the interest rates in this table to reflect the times and the individual's own financial situation.

The input table for the B matrix in GAMS is

```
Table b(n,m)  control vector matrix

       Xbe   Xbc  Xbcc  Xbsl  Xec  Xecc  Xesl

Sb      -1    -1   -1    -1

Se       1                    -1    -1    -1

Sc             1               1

Scc                -1               -1

Ssl                       -1                -1

+

       Xcacc Xcsl Xccsl

Sb

Se

Sc       -1   -1
```

```
Scc      -1           1

Ssl           -1     -1
```

This table is too wide to fit on a single page so the + symbol is used between the two parts table in GAMS to indicate that additional columns are input in a second set of rows.

Consider first only the first four columns of the B matrix, which are all transfers out of the bond account. The first two columns (Xbe and Xbc) are transfers to other assets, that is, bonds to equities and bonds to the checking account. Thus there is a –1 in the bonds row and a +1 in the equities and the checking account rows. The next two columns ($Xbcc$ and $Xbsl$) are transfers from an asset account (bonds) to liability accounts (credit card and student loan, respectively) so there is still a –1 in the bond row. However, –1 also still appears in the credit card and student loan rows since these transfers have the effect of decreasing the amount of credit card debt or of student loan debt through the action of selling bonds to pay off some of these loan amounts.

The input table for the C matrix in GAMS is

```
Table c(n,k)  exogenous vector matrix

        Wa      Le      Sh

Sb

Se

Sc      1       -1       1

Scc

Ssl
```

The only entries in this matrix are in the checking account row since wages and scholarships are deposited in this account and living expenses are withdrawn from it. The criterion function priorities (penalty matrices) are input next. The matrix for the priorities for the state vectors for all periods other than the terminal period W is

Table w(n,np) state vector matrix penalty matrix

	Sb	Se	Sc	Scc	Ssl
Sb	100				
Se		100			
Sc			400		
Scc				200	
Ssl					0

This is a diagonal $n \times n$ matrix; however, the set n and its alias np are used. This is not essential here, but it makes it easier to understand the notation that is used later in specifying the criterion function in GAMS. Since the priority for the checking account is set high at 400, one would expect that in the solution the checking account state variable Sc tracks more closely to its desired value $S\tilde{c}$ than other state variables to their respective desired values.

Next comes the input for the W_N matrix that is the state variable priority matrix for the terminal period N:DKDavid Kendrick1721258594In View: Print Layout mode there is a spurious carriage return in the *Table wn(n,np)* ... line. In this case I was able to take care of the problem by decreasing the indentation on the left in that one line.

Table wn(n,np) terminal state vector matrix penalty matrix

	Sb	Se	Sc	Scc	Ssl
Sb	200				
Se		200			
Sc			800		
Scc				200	
Ssl					1

These values are set twice as high as the priorities for the state variables in all other time periods. This is followed by the input for the Λ matrix that is the control variable priority matrix for all time periods:

Table lambda(m,mp) lambda matrix

```
       Xbe  Xbc  Xbcc  Xbsl  Xec  Xecc Xesl

Xbe    20

Xbc          1

Xbcc               20

Xbsl                    20

Xec                           20

Xecc                                20

Xesl                                     20

+

         Xcacc     Xcsl     Xccsl

Xcacc      1

Xcsl                 1

Xccsl                         20
```

All of these priorities are set to 20 except for those for transfers from the bond account to the checking account, from the checking account to the credit card account, and from the checking account to the student loan account. Thus these three transfers are permitted to deviate more from than their desired paths than are the other transfers.

Since the desired path for the state vector$\tilde{x}_t$ (x_t with a tilde over it) is time varying it can be conveniently input with a table statement:DKDavid Kendrick1721258574Here there were several problem lines with unwanted carriage returns. However, I was able to take care of the problem by changing the font size from Courier 12 to Courier 10.

Table xtilde(n,t) state vector desired paths

\[	2000	2001	2002	2003	2004\]
Sb					
Se					
Sc	1000	1000	1000	1000	1000
Scc	2000	2000	2000	2000	2000
Ssl					

Recall the GAMS convention that a blank input in a table is treated as a zero. Therefore the desired path for bonds, equities, and student loans are all set to zero. We want the checking account to hold steady at about \$1,000 and the student's credit card debt to hold at around \$2,000. The desired path for the control vector$\tilde{u}_t$ is time varying, so it can likewise be input with a table statement.

Table utilde(m,t) control vector desired paths

	2000	2001	2002	2003
Xbe				
Xbc				
Xbcc				
Xbsl				
Xec				
Xecc				
Xesl				

```
Xcacc

Xcsl

Xccsl
```

Since this table in entirely blank the desired values for all the transfers in all time periods are set to zero. After the matrices in the systems equations and criterion function are input with table statements, the next step is to input the initial period values of the state vector. Since this is a vector it can be input with a parameter statement:

```
Parameter

       xinit(n)    initial value   /

   Sb 4000

   Se    0

   Sc 1000

   Scc   0

   Ssl   0 /
```

As a first approximation, one can think of the parameter keyword in GAMS as the way to input a vector of data and the table keyword as the way to input a matrix. Thus the xinit(n) parameter was used earlier to input the vector that contains the initial values of the state variables.

The student begins, say, with $4,000 in bonds, no equities, and $1,000 in a checking account. Moreover, he or she does not initially have any credit card debt or student loan debt. This vector is particularly useful in the experiments with this model since the most obvious thing to do to tailor the model to an individual's personal situation is to change the initial values for the state variables.

Next comes the input for the exogenous variables $z_t$that are time varying. Since this is a vector that changes over time it can be input with a table statement:

```
 Table z(k,t)  exogenous variables
```

```
       2000        2001         2002        2003        2004
Wa    15000       15000        15000       15000       15000

Le    20000       20000        20000       20000       20000

Sh        0           0            0           0           0
```

The student has wages from a part-time job of $15,000 a year, living expenses of $20,000 a year, and no scholarship help, so he or she must borrow approximately $5,000 a year or draw down assets. Like the initial conditions, this table is an obvious place for tailoring the model to an individual either by altering the wages, living expenses, and scholarship aid over time or inputting a pattern more closely related to the individual's own situation with respect to these exogenous variables.[1]

The variables are the next thing to be assigned in the GAMS program:

```
Variables

 u(m,t)    control variable

     j         criterion ;

Positive Variables

 x(n,t)   state variable ;
```

Aside from the criterion variable j the only two sets of variables in the model are the control variables u and the state variables x. The control variables can be either positive or negative. For example, if the variable Xbc is positive it is a transfer from the bond account to the checking account and if it is negative it is a transfer from the checking account to the bond account. On the other hand, the state variables must be positive. For example Scc is a liability account, namely credit card debt. If this amount were negative it would mean that the student was lending money to the credit card company. While some students might like to do that at 13 percent, it is unlikely that the credit card company would be willing to enter into such a deal. Therefore the restriction that the state variables must be positive is imposed in GAMS with the key words Positive Variables.

Next the equations are declared in GAMS with the statements

[1]Thanks to one or our students, Vivek Shah, for helping to develop the time-varying exogenous variable version of the thrift model.

```
 Equations    criterion       criterion definition

                stateq(n,t)     state equation ;
```

So the only sets of equations in this model are the single equation for the criterion function and the $n \times t$ state equations. Since there are five state variables and five time periods the model has twenty-five state equations.

Next the equations are defined, beginning with the criterion function. Recall from equation (19) that this equation is in three parts: the state variables for the terminal period, the state variables for all other time periods, and the control variables for all other time periods, that is,

$$J = \frac{1}{2}\left(x_N - \tilde{x}_N\right)' W_N \left(x_N - \tilde{x}_N\right) + \frac{1}{2}\sum_{t=0}^{N-1}\left[\left(x_t - \tilde{x}_t\right)' W \left(x_t - \tilde{x}_t\right) + \left(u_t - \tilde{u}_t\right)' \Lambda \left(u_t - \tilde{u}_t\right)\right]$$

[equation (19)]

Consider for the moment only the first part, that is, the state variables in the terminal time period. This can be written with indices, rather than in vector-matrix form, as

$$J = \frac{1}{2}\sum_{i \in I}\sum_{j \in J}\left(x_{iN} - \tilde{x}_{iN}\right) w_{iN}\left(x_{jN} - \tilde{x}_{jN}\right) \tag{27}$$

This, in turn, can be represented in GAMS with the statementDKDavid Kendrick1721258578I changed the font size from Courier 12 to Courier 10 in order to eliminate the spurious carriage return in the last line.

```
criterion..

j =e=

 .5*sum( (tz,n,np),

 (x(n,tz) - xtilde(n,tz))*wn(n,np)*(x(np,tz) - xtilde(np,tz)) )
```

This code begins with the name of the equation, criterion, and the two dots (..) following the name signal to the GAMS compiler that the name has been completed and the equation itself is to follow.

The sum in the mathematics in equation (27) is over the two sets I and J, whereas the sum in GAMS is over three sets (tz,n,np). Since the set tz in GAMS has only a single element, namely the terminal period N, in fact this sum in GAMS is really only over two sets. Recall that n is the set for the state variables, which are the stocks of bonds, equities, checking account, credit card account, and student loans. Furthermore, the set np in GAMS is the alias of the set n, that is, it is a copy of the set.

The second part of the criterion function, namely the state variable for all periods other than the terminal period, is written in mathematics with indices as

$$\frac{1}{2}\sum_{t\in Tu}\sum_{i\in I}\sum_{j\in J}\left(x_{it}-\tilde{x}_{it}\right)\,w_{it}\left(x_{jt}-\tilde{x}_{jt}\right) \tag{28}$$

where

$$Tu = \text{set of all time periods except the terminal period}$$

and this is written in GAMS asDKDavid Kendrick1721258579Same problem – same solution in the 2 lines of code below. I changed the font size from Courier 12 to Courier 10 to remove the spurious carriage return.

```
+ .5*sum( (tu,n,np),

  (x(n,tu) - xtilde(n,tu))*w(n,np)*(x(np,tu) - xtilde(np,tu)) )
```

The sum here is indeed over the three sets, namely, tu, the set of all time periods other than the terminal period, and n and np, the state variable set and its alias.

The final piece of the criterion function is written in mathematics with indices as

$$\frac{1}{2}\sum_{t\in Tu}\sum_{i\in I}\sum_{j\in J}\left(u_{it}-\tilde{u}_{it}\right)\,\lambda_{it}\left(u_{jt}-\tilde{u}_{jt}\right) \tag{29}$$

and in GAMS asDKDavid Kendrick1721258580Same problem – same solution in the two lines of code below, i.e. change of font size.

```
  + .5*sum( (tu,m,mp),

   (u(m,tu) - utilde(m,tu))*lambda(m,mp)*(u(mp,tu) - utilde(mp,tu)) ) ;
```

Thus this sum is over both the time period set tu and the control variable set m and its alias mp. The only equations in the model apart from the criterion function are those in the set of system equations. Recall that these equations are written mathematically as

$x_{t+1} = Ax_t + Bu_t + Cz_t$ [equation (20)]

In GAMS they are given as

```
stateq(n,t+1)..

 x(n,t+1) =e=

  sum(np,(a(n,np)*x(np,t))) +

  sum(m, (b(n,m)*u(m,t))) +

  sum(k, (c(n,k)*z(k,t)));
```

The name of the equation in GAMS is stateq and it is defined for the sets n and t+1. Recall that the set t is

$$T = \{2000, 2001, 2002, 2003, 2004\} \tag{30}$$

Then the set t+1 in GAMS is defined as the set t less the first element in the set, namely

$$\{2001, 2002, 2003, 2004\} \tag{31}$$

Thus stateq is defined over all time periods in the model except for the first one. Finally it is necessary to specify the initial conditions for the state variables of the model with the GAMS statement

```
 x.fx(n,ti) = xinit(n);
```

The suffix .fx is used as an abbreviation for *fixed*. In this statement then the state vector x is fixed in the initial period ti to the values in the vector xinit, which is the parameter vector that contains the initial conditions for the model. Though it is not shown in the present version of the model, upper bounds on the credit card and student loan accounts can be included in GAMS statement. This is done at the end of the equations and before the Solve statement with upper bounds on variables. An example of this is as follows:

```
x.up('Scc',t)=5000;

x.up('Ssl',t)=7000;
```

The `x.up` means the upper bound for the variable `x`. So `x.up('Scc',t)` is the upper bound on the credit card and the `x.up('Ssl',t)` is the upper bound on the student loan. One can change these bounds to fit his or her own financial situation.

Next a name is assigned to the model along with an indication of which equations are included in the model. In this case all of the equations are included:

```
Model track  /all/ ;
```

This is followed by a statement directing that the model be solved with a nonlinear programming solver by minimizing j, the criterion value:

```
Solve track minimizing j using nlp ;
```

For an introduction to nonlinear optimization solvers see Appendix B and for a discussion of the stacking method in GAMS that is used for an indexed model like this one see Appendix C. Finally a table of the results is obtained by using the statement

```
Display x.l, u.l ;
```

The `suffix.l` on the variables x and u is not the number 1 but rather the letter l and is used to indicate the activity level of the variable. Though it is not shown in the present model it is also possible to solve the model by maximizing terminal wealth.

4 Results

As was discussed earlier, Appendix D contains instructions on running GAMS. Recall from that discussion that examining the results from a GAMS run can seem complicated at first because the output files contain a substantial amount of information about the structure of the model and its solution. However, it is simple enough to jump around in the file to examine the key parts.

First locate the Solve Summary part of the output. To do this search in the editor for the string SOLVER STATUS. When you do so you see a section of the output that looks like

```
               S O L V E      S U M M A R Y

     MODEL   track                  OBJECTIVE  j
     TYPE    NLP                 DIRECTION  MINIMIZE

     SOLVER  CONOPT              FROM LINE  166

**** SOLVER STATUS     1 NORMAL COMPLETION

**** MODEL STATUS      2 LOCALLY OPTIMAL

**** OBJECTIVE VALUE       1377722382.8446
```

As we suggested previously, each time you solve a GAMS model you should check this section of the output afterward to be sure that the model was solved successfully. The words `NORMAL COMPLETION` here indicate that that is the case. If the solution procedure was not successful you find words like `INFEASIBLE` or `UNBOUNDED`.

Next skip down the output across the sections labeled —- `EQU` until you get to the section labeled —- `VAR x`state variable, which looks like

```
---- VAR x  state variable

            LOWER     LEVEL     UPPER    MARGINAL

Sb .2000  4000.000  4000.000  4000.000 4.2771E+5

Sb .2001      .      463.607     +INF        .

Sb .2002      .       17.262     +INF  -1.142E-9

Sb .2003      .        2.981     +INF  -3.220E-9

Sb .2004      .         .        +INF  19101.665

Se .2000      .         .          .   24584.643

Se .2001      .      405.341     +INF       EPS

Se .2002      .       39.804     +INF  -1.291E-9

Se .2003      .       12.292     +INF  -2.918E-9
Se .2004        .         .        +INF  18069.230

Sc .2000  1000.000  1000.000  1000.000 23323.625

Sc .2001        .     1107.581     +INF  -1.062E-9

Sc .2002        .     1001.230     +INF  -1.495E-9

Sc .2003        .      998.805     +INF  -3.380E-9

Sc .2004        .      975.797     +INF       EPS

Scc.2000        .         .         .    -4.280E+5

Scc.2001        .     1766.529     +INF  -9.31E-10

Scc.2002        .     1984.219     +INF        .

Scc.2003        .     1991.238     +INF        .

Scc.2004        .     2096.163     +INF       EPS

Ssl.2000        .         .         .    -2.421E+4

Ssl.2001        .         .        +INF  39980.703

Ssl.2002        .     4018.936     +INF  1.1896E-9

Ssl.2003        .     9371.548     +INF  3.2469E-9

Ssl.2004        .    14850.699     +INF        .
```

The interesting part here is the activity level of the shipment variables x in the column labeled LEVEL. This shows, among other things, that there was $4,000 in bonds in time period 2000 and $463 in bonds in time period 2001. This is the solution of the model that we were looking for. These same results are shown a little further down in the output in a section labeled —169 VARIABLE x.L state variable, which is the result of the display statement in the GAMS input. That output is

```
----    169 VARIABLE  x.L  state variable
```

```
\[             2000         2001         2002         2003          2004\]
Sb      4000.000      463.607       17.262        2.981

 Se                    405.341       39.804       12.292

 Sc       1000.000     1107.581     1001.230      998.805      975.797

 Scc                   1766.529     1984.219     1991.238     2096.163

 Ssl                                4018.936     9371.548    14850.699
```

This table is somewhat easier to read than the default output, so you can see the reason that most GAMS input files end with a series of display statements. These tables are easily found since they are at the end of the long GAMS output so the user can quickly scroll to the bottom of the file and find the key results. However, they will be there only if you remember to add a display statement at the end of the GAMS input statement. So in summary, when looking at the GAMS output you should first check to be sure that the problem was solved satisfactorily. Then focus on the variables section.

Recall that the student starts with \$4,000 in bonds and \$1,000 in a checking account and has \$15,000 a year in wages and \$20,000 in living expenses. In addition, the desired paths for the checking and credit card accounts are \$1,000 and \$2,000, respectively.

As the table of state variable results over time above shows, the bond account is drawn down in the first two periods and the student also borrows roughly \$2,000 on her credit card. Then in the third time period borrowing begins from the student loan account and reaches about \$15,000 by the last period. So in order to finance the \$5,000 shortfall each year over the four-year period the student cashes in \$4,000 in bonds and borrows \$2,000 on her credit card and about \$15,000 from the student loan fund. Meanwhile she continues to hold about \$1,000 in her checking account in all time periods.

The transfers that are necessary to accomplish these results are shown in the following control variable time paths:

```
 ----    169 VARIABLE  u.L  control variable

\[               2000         2001         2002         2003\]
```

```
Xbe         201.984    -124.453     -55.584     -51.622

 Xbc       3296.649     721.808     275.338     260.756

 Xbcc        82.078      20.062      11.086       6.544

 Xbsl       155.683    -147.892    -215.696    -212.548

 Xec        -37.151     160.543      69.351      64.660

 Xecc      -119.906     144.515      66.670      58.166

 Xesl       -46.301     -23.439    -160.112    -160.927

 Xcacc    -1655.095    -320.572     -53.610    -129.867

 Xcsl      -182.988   -3679.651   -4589.264   -4511.721

 Xccsl       73.605    -167.954    -226.783    -219.093
```

There is a transfer of $3,296 from the bond account to the checking account, Xbc, in the first time period followed by a transfer of $721 in the second period. There is also a negative transfer of about –$1,600 from the checking account to the credit card account, Xcacc, in the first time period, which is actually a transfer of about $1,600 from the credit card account to the checking account. This in turn is followed by a similar transfer of about $300 in the second time period.

The borrowing from the student loan fund begins in the second time period when about $3,600 is transferred from the student loan account to the checking account via the variable Xcsl. This is followed by transfers of approximately $4,500 of a similar nature in the third and fourth time periods.

5 Experiments

The most useful experiment to do with this model is to use it to take a rough look at your own finances during college and graduate school. The most important steps to accomplish this are to change the initial conditions in xinit and the wages, living expenses, and scholarship aid in the exogenous variables z. Moreover, the interest rates are likely to be different than those used in the example and should be modified to be realistic.

Finally you may have very different desired paths for the state variables. For example, you may want to keep the bond account constant over the time horizon covered by the model or want to limit credit borrowing to a smaller amount than was used in the foregoing model.

There are other interesting experiments you can do simply by including bounds on the variables. For example, you can put a lower bound on the checking account. Some students have accounts in which they are supposed to keep at least a minimum balance, namely $800, so you can place a lower bound of $800 on the checking account. In GAMS code the bound would look like

```
x.lo('Sc',t)=800
```

More complicated experiments are to increase the time horizon covered by the model, say, to about ten periods and thus to cover not only years in college but the first years of employment, when paying back student debt may become a priority. Another possibility is to solve the model by maximizing terminal wealth instead of minimizing the criterion function. Some of these last three experiments require changes in the specification of the model and are more difficult; however, they are a good way to learn more about GAMS and about financial planning.

References

Judd, K. L.: 1998, *Numerical methods in economics*, MIT Press, Cambridge, Massachussets.

Miranda, M. J. and Fackler, P. L.: 2002, *Applied Computational Economics and Finance*, MIT Press, Cambridge, Massachussets.

Table 1: Input-Output

commodity		process		
	Pig iron production	Steel production pig iron intensive	Steel production scrap intensive	Rolling flat steel products
Iron ore	–1.6			
Pig iron	1.0	–0.9	–0.7	
Scrap iron		–0.2	–0.4	
Liquid steel		1.0	1.0	–1.2
Scrap				0.2
Flat steel				1.0

Appendix

A An Example U.S. Economy Database

This appendix contains the relationships in the example U.S. economy database from Kendrick (1982b).

$$\textit{test} \tag{A-1}$$

In the Input-Output relationship negative values indicate inputs and positive values outputs. Thus in the pig iron production process 1.6 tons of iron ore are used to produce 1.0 tons of pig iron. In the next column 0.9 ton of that pig iron is used along with 0.2 ton of scrap to produce 1 ton of liquid steel. The activity analysis vectors here follow in the tradition of Tjalling Koopmans (1951). Also for an introduction to use of activity analysis in economics see Kendrick (1996).

Thus a process is akin to a cook's recipe in that it provides a list of ingredients and how much is required of each as well as an indication of the final product or products. However, unlike the usual recipe for a cake, a process may have a single input and multiple outputs as, for example, the foregoing process for primary distillation in an oil refinery where crude oil input is transformed into distillate, gasoline, and jet fuel.Kendrick/Ch 5 35 06/29/08 8:57 PM

Table 2: Regional Production

	Alumina production	Aluminum production	Primary distillation	Catalytic cracking
Bauxite	–1.4			
Alumina	1.0	–1.2		
Aluminum		1.0		
Crude oil			–1.0	
Distillate			0.2	–1.0
Gasoline			0.3	0.6
Jet fuel			0.1	0.2

Table 3: Regional Production

	Auto body stamping	Auto assembly
Flat steel	–1.2	
Aluminum	–0.2	
Auto bodies	1.0	–1.0
Automobiles		1.0

Table 4: Production

commodity		process		
	Pig iron production	Steel production pig iron intensive	Steel production scrap intensive	Rolling flat steel products
Pig iron (mty)	86.8			
Liquid steel (mty)		55.5	53.0	
Scrap (mty)				18.0
Flat steel (mty)				90.0

Table 5: Regional Production

	Alumina production	Aluminum production	Primary distillation	Catalytic cracking
Alumina (mty)	20.0			
Aluminum (mty)		16.0		
Distillate (tby)			1.46	
Gasoline (tby)			2.19	2.43
Jet fuel (tby)			0.73	0.73

Table 6: Regional Production

	Auto body stamping	Auto assembly
Auto bodies (muy)	9.5	
Automobiles (muy)		9.35

Note: mty = million tons per year; tby = trillion barrels per year; muy = million units per year.

Table 7: Regional Production

productive unit		process		
	Pig iron production	Steel production pig iron intensive	Steel production scrap intensive	Rolling flat steel products
Blast furnace	1			
Steel shop		1	1	
Rolling mill				1

Table 8: Regional Production

	Alumina production	Aluminum production	Primary distillation	Catalytic cracking
Alumina plant	1			
Aluminum plant		1		
Primary still			1	
Catalytic cracker				1

@T:Table 5A.3 Capacity Use

The Capacity Use relationship simply indicates the productive unit in which each process runs. Note that substitute processes such as the two for steel production both use the same productive unit, namely the steel shop.

Note: mty = million tons per year; mbd = million barrels per day; muy = million units per year.

Table 5A.7 is really about investment, but it differs from the usual notion of investment, which is a certain number of dollars spent on a new plant or equipment. Rather the investment in the table is defined as an increment to capacity and is measured in units of the principal input or output of the productive unit. Thus the blast furnace capacity at Sparrows Point is increased by 0.5 million tons per year (an output) and the primary still at ARCO-Houston is increased by 0.1 million barrels per day (an input).

Table 9: Regional Production

	Auto body stamping	Auto assembly
Auto stamping plant	1	
Auto assembly plant		1

Table 10: Industry Composition

plant	industry
Sparrows Point	Steel
Rockdale	Aluminum
ARCO-Houston	Oil
Point Comfort	Aluminum
Inland-Gary	Steel
Lansing	Automobile

Table 11: Sector Composition

industry	sector
Steel	Primary metal
Aluminum	Primary metal
Oil	Petroleum and coal
Automobile	Transportation equipment

Table 12: Capacity 1980

productive unit	plant	capacity level	units
Blast furnace	Sparrows Point	2.0	mty
Blast furnace	Inland-Gary	2.5	mty
Steel shop	Sparrows Point	2.35	mty
Steel shop	Inland-Gary	2.8	mty
Rolling mill	Sparrows Point	1.9	mty
Rolling mill	Inland-Gary	2.4	mty
Alumina plant	Point Comfort	0.8	mty
Aluminum plant	Point Comfort	0.6	mty
Aluminum plant	Rockdale	0.5	mty
Primary still	ARCO-Houston	0.2	mbd
Catalytic cracker	ARCO-Houston	0.23	mbd
Auto stamping plant	Lansing	0.6	muy
Auto assembly line	Lansing	0.6	muy

Table 13: Increment to Capacity 1981

productive unit	plant	increment to capacity	units
Alumina plant	Point Comfort	0.5	mty
Aluminum plant	Point Comfort	0.4	mty
Auto assembly line	Lansing	0.0	muy
Auto stamping plant	Lansing	0.0	muy
Blast furnace	Sparrows Point	0.5	mty
Blast furnace	Inland-Gary	0.0	mty
Catalytic cracker	ARCO-Houston	0.12	mbd
Primary still	ARCO-Houston	0.1	mbd
Rolling mill	Sparrows Point	0.4	mty
Steel shop	Sparrows Point	0.5	mty
Aluminum plant	Rockdale	0.0	mty
Steel shop	Inland-Gary	0.0	mty
Rolling mill	Inland-Gary	0.0	mty

Table 14: Ownership

plant	corporation
Sparrows Point	United States Steel
Rockdale	ALCOA
ARCO-Houston	Atlantic Richfield Co.
Point Comfort	ALCOA
Inland-Gary	Inland Steel
Lansing	General Motors

Table 15: Plant Employees (in thousands of employees

plant			union			
	OCAW	UAW	USA	IBEW	IBT	IAM
Sparrows Point			1.2		0.3	0.05
Rockdale			0.5	0.05		
ARCO-Houston	0.4				0.01	
Point Comfort			0.7		0.2	
Inland-Gary			0.4			
Lansing		1.2				

Note: OCAW, Oil, Chemical and Atomic Workers; UAW, United Auto Workers; USA, United Steel Workers of America; IBEW, International Brotherhood of Electrical Workers; IBT, International Brotherhood of Teamsters; IAM, International Association of Machinists.

Which relationships share a common domain?

Plant Location and City Location are linked by *city*

City Location and State Location and

State Governor are linked by *state*

State Governor and Party Affiliation are linked by *governor*

Ownership and Plant are linked by *corporation*

Table 16: Plant Location

plant	city
Sparrows Point	Sparrows Point
Rockdale	Rockdale
ARCO-Houston	Houston
Point Comfort	Point Comfort
Inland-Gary	Gary
Lansing	Lansing

Table 17: City Location

city	state
Sparrows Point	Maryland
Rockdale	Texas
Houston	Texas
Point Comfort	Texas
Gary	Indiana
Lansing	Michigan

Table 18: State Location

state	region
Maryland	East Coast
Texas	Gulf Coast
Indiana	Mid-West
Michigan	Mid-West

Table 19: State Governors

state	governor
Maryland	Harry Hughes
Texas	William P. Clements, Jr.
Indiana	Otis R. Bowen
Michigan	William G. Milliken

Table 20: Party Affiliation

governor	party
Harry Hughes	Democrat
William P. Clements, Jr.	Republican
Otis R. Bowen	Republican
William G. Milliken	Republican

Industry Composition and
Sector Composition are linked by *industry*
Input-Output and Production are linked by *process* and *commodity*
Production and Capacity Use are linked by *process*
Capacity Use and Capacity are linked by *productive unit*
Increment to Capacity and Capacity are linked by *productive unit* and *plant*
Capacity and Plant Employees are linked by *plant*
Increment to Capacity and
Plant Employees are linked by *plant*
Industry Composition and
Plant Employees are linked by *plant*

B Nonlinear Optimization Solvers

Solving nonlinear optimization problems usually requires the use of numerical methods. In general, those methods consist of a *smart* trial-and-error algorithm that is a finite sequence of computational steps designed to look for convergence to a solution. There is a variety of algorithms to solve nonlinear problems. Some of them are global methods, in the sense that they perform a parallel exploration of many regions of the optimization space, for example, the genetic algorithm. Other are local, as they tend to focus on the exploration of a particular region of the optimization space. We introduce here two of the most popular local methods—the gradient and the Newton methods—used by the solvers in Excel, **GAMS**, and **MATLAB**, but before introducing them, we give with a simple example.

Suppose that we are trying to find the maximum of a nonlinear function

$$y = f(x) \tag{B-1}$$

such as the one represented in Figure 1.

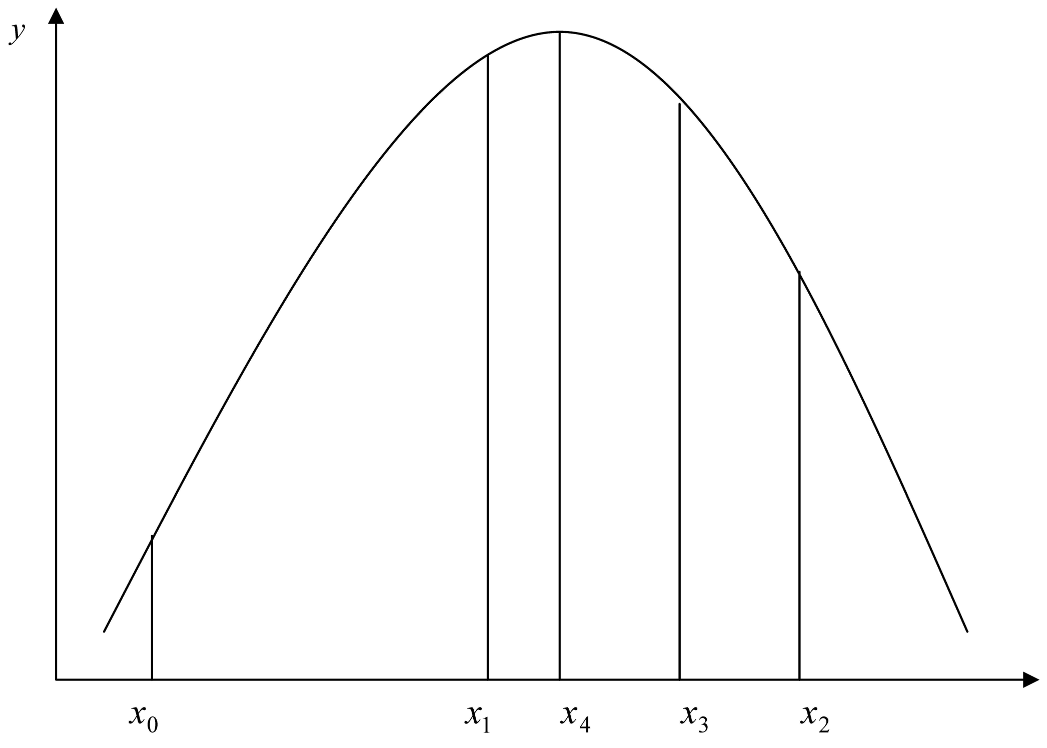
A simple and very rudimentary algorithm to find the solution might be as follows: We choose an arbitrary initial value for x, such as x_0 in Figure 1, and compute the corresponding $y_0 = f(x_0)$. We then increase that value by a constant magnitude h (we name this magnitude the *search step*) that we also choose in an arbitrary way. For the new value of x, that is x_1, we compute the corresponding value of

$$y_1 = f(x_1) = f(x_0 + h) \tag{B-2}$$

and compare this value to the one obtained in the previous step. We continue to do this as long as the differences between two successive values of y are positive (negative for a minimization problem). As soon as we compute a difference with a negative sign (in Figure F.1 this would correspond to x_2), we reverse the direction of the search. We begin to move in the opposite direction along x (i.e., subtracting h from x) and use a smaller value for h than the one we were using while we moved in the opposite direction. We continue like this until we again find a difference between two successive values of y that is negative, at which point we again reverse the direction of the search and further reduce the size of h, and so on. We stop when the difference between two successive values of y falls below a preestablished tolerance limit.

The gradient and the Newton methods are iterative like the one just presented. However, they exploit local information about the form of the function; that is,

Figure 1: A nonlinear function.



they use the function's derivatives. To illustrate this we change to a multivariate example. In this case we use the following equation to obtain each new value of the vector x:

$$x_{n+1} = x_n + h\Delta x \tag{B-3}$$

where h is the search step—now always a positive value—and Δx is the direction of change, which, as we will see, is determined by the function's derivatives.

The gradient method uses the first derivative or gradient, which gives us information about how the function changes in the neighborhood of a given point. Its basic framework is the well-known first-order Taylor approximation,

$$f(x_{n+1}) \cong f(x_n) + h\nabla f(x_n)\Delta x \tag{B-4}$$

where $\nabla f(x_n)$ is the gradient vector. Note that since h is supposed to be positive, the best direction of motion is

$$\Delta x = \nabla f(x_n) \tag{B-5}$$

for a maximization problem, since

$$f(x_{n+1}) \cong f(x_n) + h\nabla(f(x_n))^2 > f(x_n) \tag{B-6}$$

In addition, for a minimization problem

$$\Delta x = -\nabla f(x_0) \tag{B-7}$$

since

$$f(x_{n+1}) \cong f(x_n) - h\nabla(f(x_n))^2 < f(x_n) \tag{B-8}$$

The basic framework of the Newton method is the second-order Taylor approximation

$$f(x_{n+1}) \cong f(x_n) + h\nabla f(x_n)\Delta x + \frac{h}{2}\Delta x' H(x_n)\Delta x \tag{B-9}$$

where $H(x_0)$ is the second-order derivative or Hessian, which tells us how the slope of the function changes in a neighborhood of a given point.

Assuming the Taylor expansion of a function f is a good global approximation of that function, we approximate the optimum value of f by optimizing its Taylor expansion. In our case, this is equivalent to saying that to determine the best direction of motion Δx we have to optimize the expression (f.9). Differentiating (f.9) with respect to Δx, setting the result equal to zero, and solving for Δx, we obtain

$$\Delta x = -\frac{\nabla f(x_n)}{H(x_n)} \tag{B-10}$$

which is the best direction of motion for the Newton method.

Sometimes iterative methods like the ones presented here do not converge to a solution after a finite number of iterations. This problem can be overcome by changing the maximum number of iterations, or the size of the search step, or the tolerance limit, or the initial value of the search. Most solvers allow you to change these parameters.

Note also, as is the general case for numerical methods dealing with nonlinear optimization problems, that if there is more than one local optimum we will find only one of them. Thus, we never know for sure if the optimum we reached was a local or a global one. A rough way of dealing with this difficulty is to solve the problem providing the algorithm with alternative initial values of the search.

In this appendix we presented three numerical methods of increasing complexity. Of course, the more complex ones make use of more information, thus reducing, in general, the number of steps needed to achieve convergence. However, those steps become more complex, since they require the computation of a gradient or a Hessian. Then there are trade-offs to be evaluated when choosing a solution method.

There are additional methods for solving nonlinear problems numerically—for example, the conjugate gradient method, the penalty function method, and the sequential quadratic programming—a number of which extend, combine, or mimic the ones introduced here. For a comprehensive presentation refer to Judd (1998) and Miranda and Fackler (2002). The Excel solver uses a conjugate gradient method or a Newton method. **GAMS** uses a variety of methods, depending on what you choose or have set up as the default nonlinear solver. The **MATLAB** solver used in Chapter 7 and invoked by the fmincon function uses a sequential quadratic programming method. For details on the specific methods used by **Excel**, **GAMS**, and **MATLAB** refer to their corresponding user's and solver's manuals.

C The Stacking Method in GAMS

As a compact way of expressing a multiequation model, GAMS allows us to write indexed equations. As seen in several of the chapters in this book, those indices may represent commodities, locations, time periods, and so on.

For example, the equations corresponding to a problem such as

$$\max J = \sum_{i=0}^{2} w_1 x_i + w_2 y_i \tag{C-1}$$

subject to the constraints

$$a_{11} x_i + a_{12} y_i = b_1 \tag{C-2}$$
$$a_{21} x_i + a_{22} y_i = b_2 \tag{C-3}$$

can be represented in GAMS as

```
eqj..       j =e= sum(i, w1 * x(i) + w2 * y(i));
eq1(i)..    a11 * x(i) + a12 * y(i)) =e= b1;
eq2(i)..    a21 * x(i) + a22 * y(i)) =e=  b2;
```

When the index set is $i = \{0, 1, 2\}$ the model is expanded and stacked in the following way:

```
j =e= w1*x(0) + w2*y(0) + w1*x(1) + w2*y(1) + w1*x(2) + w2*y(2)
eq1(0)..    a11 * x(0) + a12 * y(0)) =e= b1;
eq2(0)..    a21 * x(0) + a22 * y(0)) =e= b2;
eq1(1)..    a11 * x(1) + a12 * y(1)) =e= b1;
eq2(1)..    a21 * x(1) + a22 * y(1)) =e= b2;
eq1(2)..    a11 * x(2) + a12 * y(2)) =e= b1;
eq2(2)..    a21 * x(2) + a22 * y(2)) =e= b2;
```

Note that previously we had a model with an objective function and two indexed equations and two variables [x(i) and y(i)] and now we have a model with one objective function, six equations, and six variables [x(0), x(refeq1), x(refeq2), y(0), y(refeq1) and y(refeq2)]. Thus, before solving the model, GAMS transforms a model of n indexed equations into one of *n x card* equations plus the objective function, where *card* indicates the number of elements in the index set. If the index denotes time periods, this is equivalent to transforming a dynamic model with n indexed equations and t time periods into an equivalent static model of $n \times t$ equations plus the objective function.

When, as in Chapters 8 and 13, we are interested in solving a system of equations and not an optimization problem, we just set the objective function equal to any constant value (i.e., j =e= 0;). Thus, when executing the corresponding solver statement,

```
solve model maximizing j using nlp;
```

GAMS expands and stacks the system of equations and it solves it as a by-product of a pseudo-optimization.

D Running GAMS

Appendix D provides the details for running the GAMS software on a PC. In order to use GAMS with other input files, substitute the appropriate file name for trnsport.gms in the following. For help and information about obtaining GAMS go to the GAMS Development Corporation web site at

`http://www.gams.com`

There is a student version of the software that can be downloaded and that can solve all or almost all of the models used in this book. It the model is too large, usually a small change in the number of time periods or some other set is sufficient to reduce the size so that it runs on the student version.

- Go to the book web site at

  `http://www.eco.utexas.edu/compeco`

  and to the *Input Files for Chapters in the Book* section of the web site. Right click on the `trnsport.gms` file name and select the `Save Target As` ... option in order to save the file in your preferred directory.

- Choose Programs from the Start menu and then choose GAMS and gamside. Choose Open from the File menu, navigate to the trnsport.gms file, and open it for editing. Note in the complete GAMS statement of the model that, as is the usual case in GAMS, the model is defined in steps:

  1. first the sets
  2. then the parameters
  3. then the variables
  4. then the equations
  5. and finally the model and solve statements.

- Solve the model by choosing Run from the File menu and then check the solution log to be sure that you have

  ```
  SOLVER STATUS: 1 NORMAL COMPLETION
  ```

  and

```
MODEL STATUS: 1 OPTIMAL
```

Then close the log file window.

- Click on the trnsport.lst file window and scroll through this listing file to see the solution. Note that the `*.lst` file extension used here is an abbreviation for a *listing* of the output file. Note that the GAMS output has the following structure:

  Echo Print shows a listing of the input file with the line numbers added.

  Error Messages In the case of errors in the input file, they will be signaled by GAMS with `****` on the leftmost part of the corresponding line of input where the error was found, and with `$number` just below the part of the line of input where the error is located, where `number` contains a specific error code. Then, at the end of the list of the input file, GAMS displays the explanation of each of the error codes found.

  `Equation Listing` Shows each equation of the model with the corresponding values for sets, scalars, and parameters.

  `Column Listing` Shows a list of the equations' individual coefficients classified by columns.

  `Model Statistics` Shows information such as model number of equations, number of variables, and so on.

  `Solve Summary` Shows information such as solver and model status at the end of the GAMS run, and so on.

  `Solution Listing` Shows the solution values for each equation and variable in the model. Each solution value is listed with four pieces of information, where a dot `.` means a value of zero, `EPS` a value near zero, and `+INF` and `-INF` mean plus or minus infinite, respectively:

  `LOWER (the lower bound)`

  `LEVEL (the solution level value)`

  `UPPER (the upper bound)`

  `MARGINAL` (the solution marginal value; for linear or nonlinear programming problems, it corresponds to the increase in the objective value owing to a unit increase in the corresponding constraint)

  `Report Summary` Shows the count of rows or columns that are infeasible, nonoptimal, or unbounded.