CASE STUDY

fourteen

Animating The Simplex Method

case study OVERVIEW

CS14.1 *Application Overview and Model Development*

The simplex method is one of the most popular methods to solve linear programming problems. These types of problems consist of optimizing a linear objective function subject to a set of linear constraints. In this application, we animate the simplex method to solve a user-defined linear programming problem.

CS14.1.1 Model Definition and Assumptions

The simplex method, developed by George Dantzig in 1947, maintains a basic feasible solution at every step. Given a basic feasible solution, the method first applies the optimality criteria to test the optimality of the current solution. If it does not fulfill this condition, then the algorithm performs a pivot operation to obtain another basis structure with a lower or the same cost. The simplex method repeats this process until the current basic feasible solution satisfies the optimality criteria. We will now describe the simplex algorithm using a numerical example.

Let's consider the following linear programming problem in which we seek to maximize a value written in terms of four variables. This objective is limited by three constraints and a non-negative variable requirement.

Maximize: $z = 2x_1 + x_2 + 5x_3 - 3x_4$

subject to:

$x_1 + 2x_2 + 4x_3 - x_4 \le 6$

$2x_1 + 3x_2 - x_3 + x_4 \le 12$

$x_1 + x_3 + x_4 \le 4$

$x_1, x_2, x_3, x_4 \ge 0$

The problem must first be modified to canonical form before the simplex method can be applied. The addition of slack variables and the transformation to canonical form restates the problem as follows:

Maximize: $z = 2x_1 + x_2 + 5x_3 - 3x_4 + 0x_5 + 0x_6 + 0x_7$

subject to:

$x_1 + 2x_2 + 4x_3 - x_4 + x_5 = 6$

$2x_1 + 3x_2 - x_3 + x_4 + x_6 = 12$

$x_1 + x_3 + x_4 + x_7 = 4$

$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \ge 0$

Note that there are now seven variables and three constraints and that adding the slack variables has not changed the value of the objective function or constraints. The coefficients of all of the variables in the objective function and constraints can now be written as a matrix, or tableau. Here is a representation of the initial tableau:

	z	x_1	x_2	x_3	x_4	x_5	x_6	x_7	RHS
z	1	-2	-1	-5	3	0	0	0	0
x_5	0	1	2	4	-1	1	0	0	6
x_6	0	2	3	-1	1	0	1	0	12
x_7	0	1	0	1	1	0	0	1	4

The coefficient values of the constraints' slack variables form an identity matrix. This tableau is used to perform the pivot operations for each iteration of the simplex method. These operations identify a nonbasic entering variable that has the largest negative value (for maximization problems) or the largest positive value (for minimization problems) in the objective function. The applicatoin then compares the ratios of the corresponding column coefficients to the RHS column coefficients to find the basic variable with the minimum ratio. This variable is the leaving variable. Then, the application performs pivot operations to make this column have 0 and 1 coefficient values (to become part of the identity matrix) such that the 1 coefficient value is in the row of the leaving variable. The leaving and entering variables are then switched to complete the iteration.

For example, the first iteration of the simplex method for the problem defined above selects x_3 as the entering variable since it has the largest negative coefficient in the objective function (-5). It then compares the ratios of 6/4, -12/1, and 4/1; excluding the negative ratio, the 6/4 ratio can be declared as the minimum ratio, thus identifying x_5 as the leaving variable. Pivot operations is then performed and the variables are switched to yield the resulting tableau for the first iteration:

	z	x_1	x_2	x_3	x_4	x_5	x_6	x_7	RHS
z	1	-0.75	1.5	0	1.75	1.25	0	0	7.5
x_3	0	0.25	0.5	1	-0.25	0.25	0	0	1.5
x_6	0	2.25	3.5	0	0.75	0.25	1	0	13.5
x_7	0	0.75	-0.5	0	1.25	-0.25	0	1	2.5

This application requires that the linear program is stated in canonical form. It allows the user to select the entering nonbasic variable while the minimum ration and leaving variable are determined. The application automatically checks for the unboundedness of the optimal solution. For more details on the simplex method, see *Introduction to Operations Research* by Winston.

CS14.1.2 Input

The input for this application is the following:

- Initial tableau
- Number of variables (after transformation to canonical form)
- Number of constraints
- Maximization or minimization objective
- Entering variable for each iteration

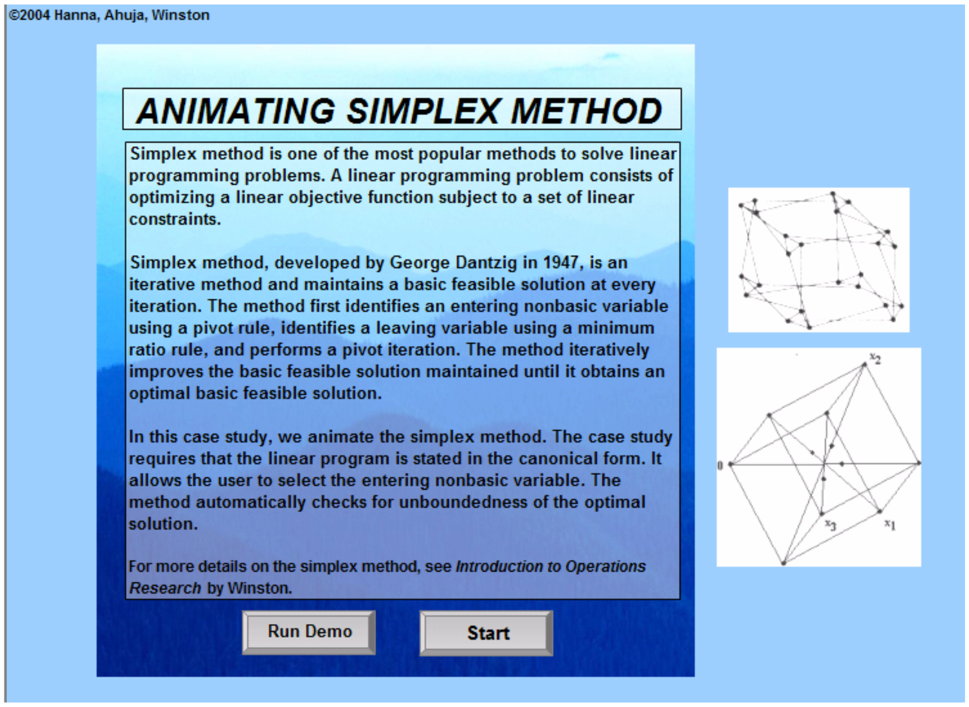
CS14.1.3 Output

The output for this application is the following:

- Leaving variable for selected entering variable
- Tableau for each iteration
- Objective function value for each iteration
- Final tableau
- Chart of change in objective function over all iterations
- Table of entering variable, leaving variable, minimum ratio, reduced cost, and objective function value for each iteration

CS14.2 *Worksheets*

This application requires four worksheets: the welcome sheet, the input sheet, the example sheet, and the report sheet. The welcome sheet contains the title, the description of the application, and the "Run Demo" and "Start" buttons. (See Figure CS14.1.) The "Run Demo" button takes the user to the input sheet and creates the initial tableau from the example sheet. (See Figure CS14.3.) The "Start" button takes the user to the input sheet to create his or her own initial tableau.


©2004 Hanna, Ahuja, Winston

ANIMATING SIMPLEX METHOD

Simplex method is one of the most popular methods to solve linear programming problems. A linear programming problem consists of optimizing a linear objective function subject to a set of linear constraints.

Simplex method, developed by George Dantzig in 1947, is an iterative method and maintains a basic feasible solution at every iteration. The method first identifies an entering nonbasic variable using a pivot rule, identifies a leaving variable using a minimum ratio rule, and performs a pivot iteration. The method iteratively improves the basic feasible solution maintained until it obtains an optimal basic feasible solution.

In this case study, we animate the simplex method. The case study requires that the linear program is stated in the canonical form. It allows the user to select the entering nonbasic variable. The method automatically checks for unboundedness of the optimal solution.

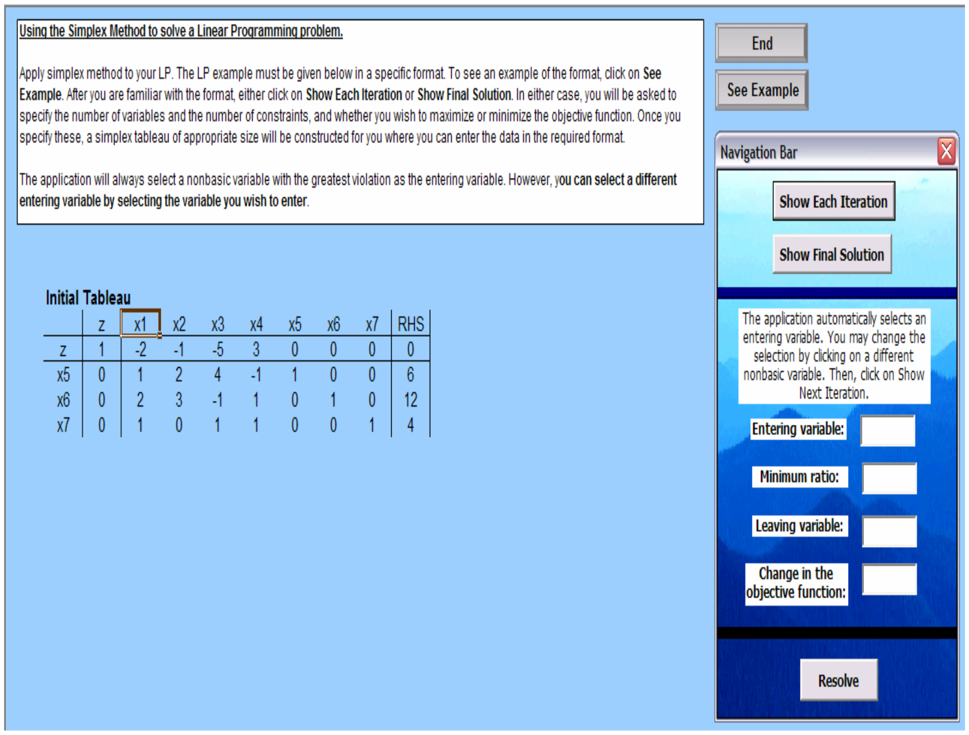
For more details on the simplex method, see *Introduction to Operations Research* by Winston.

Figure CS14.1 The welcome sheet.

The input sheet instructs the user how to create the initial tableau and animate the simplex method iterations. A navigational form, which is always available to the user on the input sheet, provides several options for performing the iterations. (We will discuss this process in detail in the next section.) The user can also view the example sheet for guidance in constructing the initial tableau by clicking the "See Example" button.

The example sheet contains an example of a linear programming problem. (See Figure CS14.3.) It reveals how to transform a problem into canonical form by adding slack variables. It also provides the initial tableau for this problem, which is the same initial tableau used for the demo option.

The report sheet displays a summary report of the iterations performed. (See Figure CS14.4.) A table lists the entering variable, the leaving variable, the minimum ratio, the reduced cost, and the objective function value for each iteration. Additionally, the sheet contains a chart of the objective function values over all the iterations.



Using the Simplex Method to solve a Linear Programming problem.

Apply simplex method to your LP. The LP example must be given below in a specific format. To see an example of the format, click on **See Example**. After you are familiar with the format, either click on **Show Each Iteration** or **Show Final Solution**. In either case, you will be asked to specify the number of variables and the number of constraints, and whether you wish to maximize or minimize the objective function. Once you specify these, a simplex tableau of appropriate size will be constructed for you where you can enter the data in the required format.

The application will always select a nonbasic variable with the greatest violation as the entering variable. However, **you can select a different entering variable by selecting the variable you wish to enter**.

Initial Tableau

	z	x1	x2	x3	x4	x5	x6	x7	RHS
z	1	-2	-1	-5	3	0	0	0	0
x5	0	1	2	4	-1	1	0	0	6
x6	0	2	3	-1	1	0	1	0	12
x7	0	1	0	1	1	0	0	1	4

The application automatically selects an entering variable. You may change the selection by clicking on a different nonbasic variable. Then, click on Show Next Iteration.

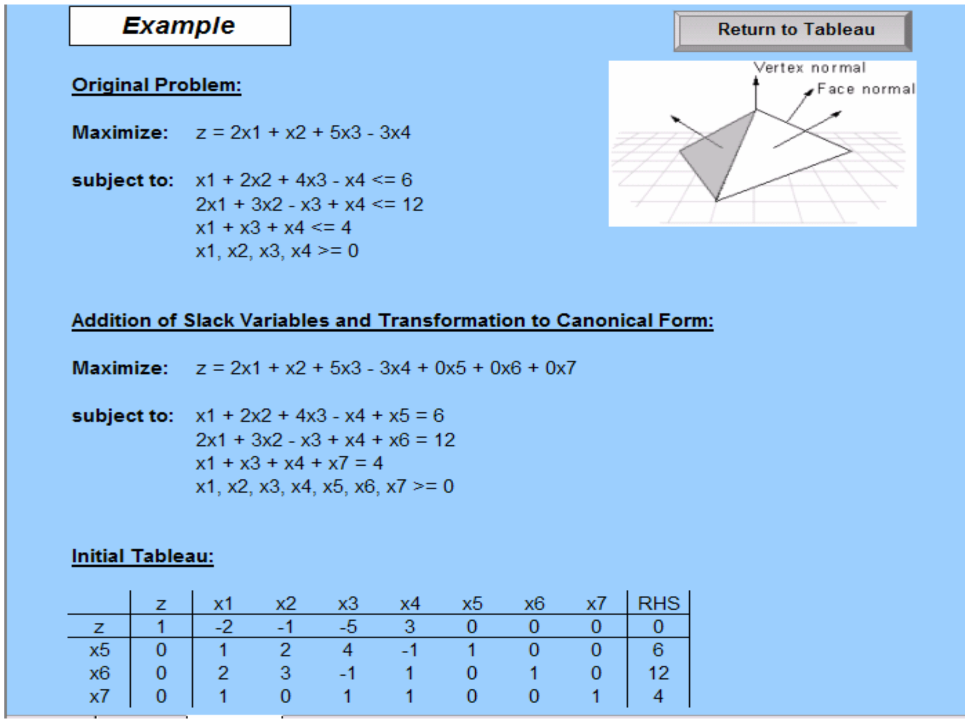
Entering variable:

Minimum ratio:

Leaving variable:

Change in the objective function:

Figure CS14.2 The input sheet.



Example

Original Problem:

Maximize: z = 2x1 + x2 + 5x3 - 3x4

subject to: x1 + 2x2 + 4x3 - x4 <= 6
2x1 + 3x2 - x3 + x4 <= 12
x1 + x3 + x4 <= 4
x1, x2, x3, x4 >= 0

Addition of Slack Variables and Transformation to Canonical Form:

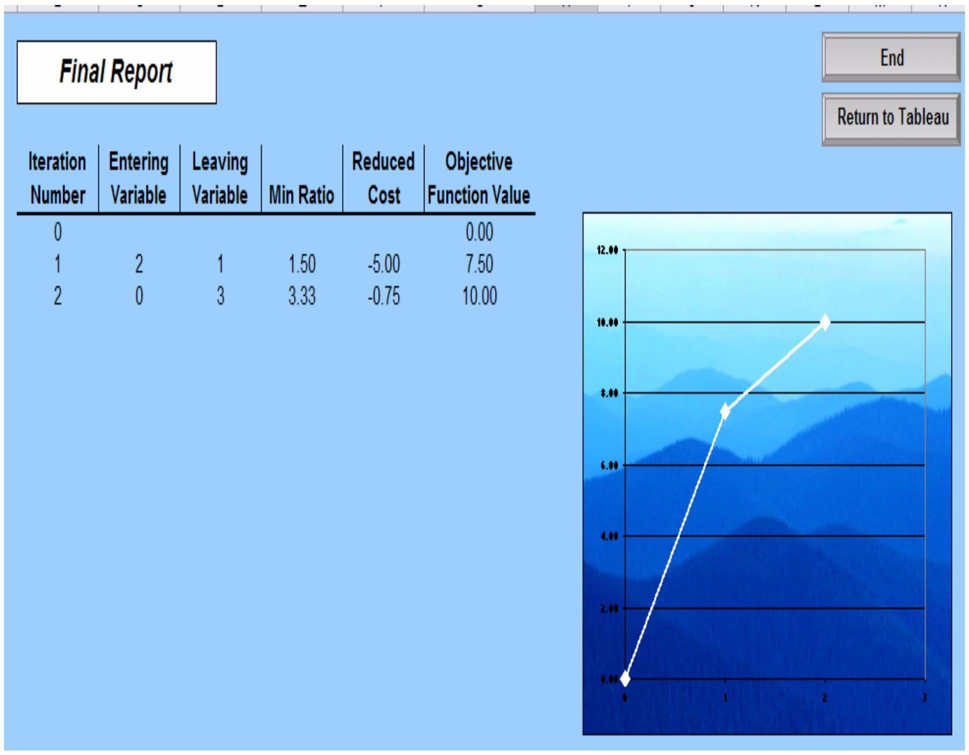
Maximize: z = 2x1 + x2 + 5x3 - 3x4 + 0x5 + 0x6 + 0x7

subject to: x1 + 2x2 + 4x3 - x4 + x5 = 6
2x1 + 3x2 - x3 + x4 + x6 = 12
x1 + x3 + x4 + x7 = 4
x1, x2, x3, x4, x5, x6, x7 >= 0

Initial Tableau:

	z	x1	x2	x3	x4	x5	x6	x7	RHS
z	1	-2	-1	-5	3	0	0	0	0
x5	0	1	2	4	-1	1	0	0	6
x6	0	2	3	-1	1	0	1	0	12
x7	0	1	0	1	1	0	0	1	4

Figure CS14.3 The example sheet.



Final Report

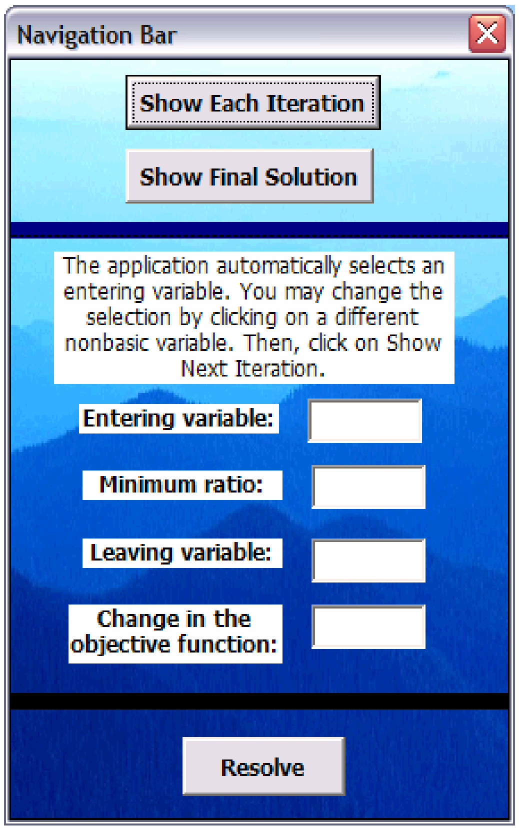
Iteration Number	Entering Variable	Leaving Variable	Min Ratio	Reduced Cost	Objective Function Value
0					0.00
1	2	1	1.50	-5.00	7.50
2	0	3	3.33	-0.75	10.00

Figure CS14.4 The report sheet.

Summary		
	Welcome sheet	Contains the application description and the "Run Demo" and "Start" buttons.
	Input sheet	Where the user provides the initial tableau and other input through the available forms; the iterations are then performed.
	Example sheet	Contains an example linear programming problem and the initial tableau; where the transformation to the canonical form occurs.
	Report sheet	Contains the summary table for each iteration and the chart of the objective function value over all the iterations

CS14.3 *User Interface*

For this application's user interface, we use navigational buttons and two user forms. The input sheet contains a navigational form that is always shown. (See Figure CS14.5.) It is a dynamic floating form with several different buttons appearing as options, which are made available to the user. When the user first arrives at the input sheet, the navigational form appears, as shown in Figure CS14.5(a). Here, the main two options available to the user are "Show Each Iteration" and "Show Final Solution." When these buttons are clicked, the user's initial tableau is checked and a second form appears.


Navigation Bar

Show Each Iteration

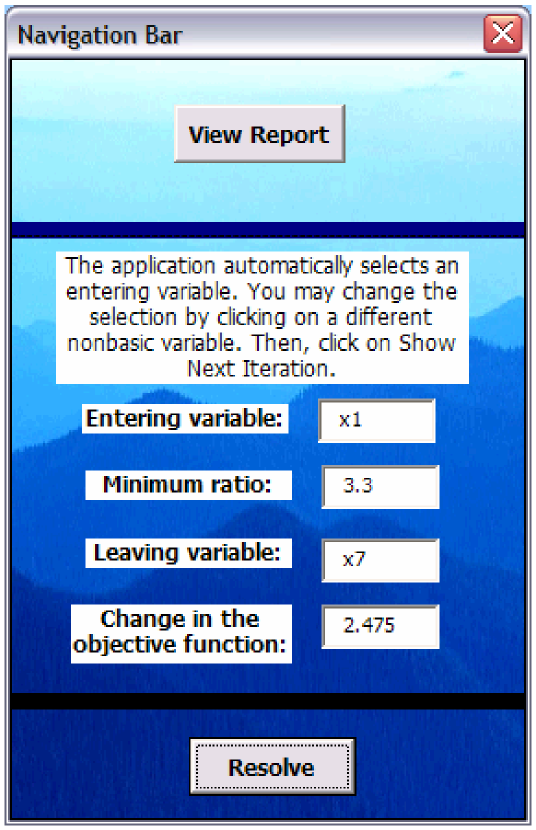
Show Final Solution

The application automatically selects an entering variable. You may change the selection by clicking on a different nonbasic variable. Then, click on Show Next Iteration.

Entering variable:	
Minimum ratio:	
Leaving variable:	
Change in the objective function:	

Resolve

Figure CS14.5 (a)


Navigation Bar

View Report

The application automatically selects an entering variable. You may change the selection by clicking on a different nonbasic variable. Then, click on Show Next Iteration.

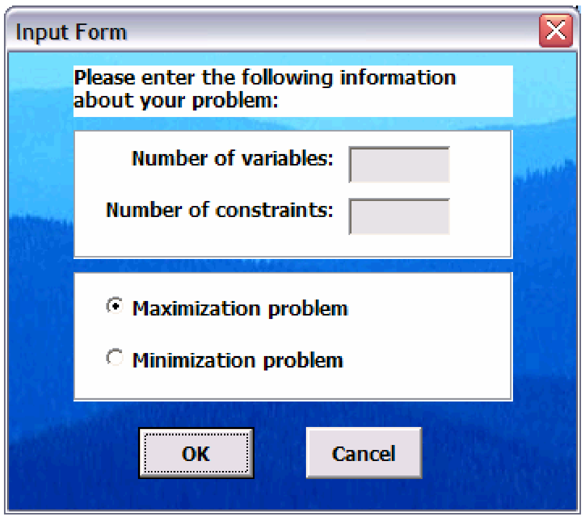
Entering variable:	x1
Minimum ratio:	3.3
Leaving variable:	x7
Change in the objective function:	2.475

Resolve

Figure CS14.5 (b)

Figure CS14.5 The navigational form.

This second form is the input form. (See Figure CS14.6.) Here, the user inputs the number of variables (including the slack variables) and the number of constraints in the problem. He or she also specifies if the problem has a maximization or minimization objective.


Input Form

Please enter the following information about your problem:

Number of variables:	
Number of constraints:	

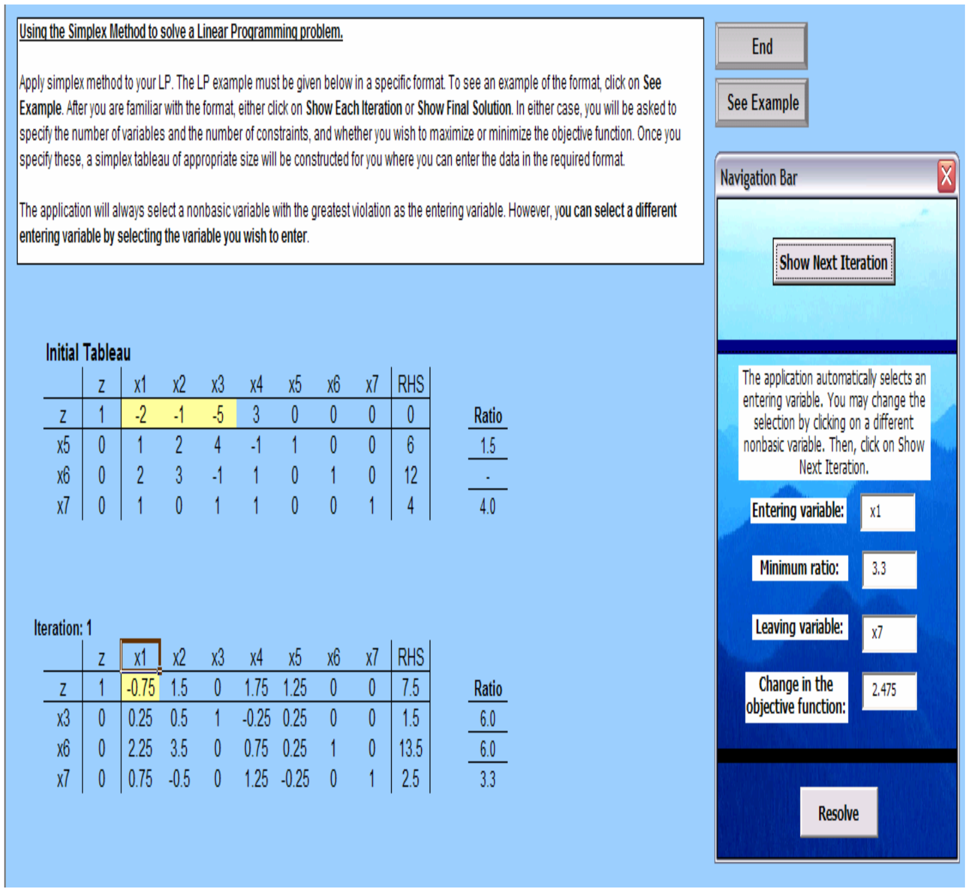
[✓ Maximization problem]

Minimization problem

OK Cancel

Figure CS14.6 The input form.

If the user selects to view each iteration, then the navigational form changes to display a “Show Next Iteration” button. As the iterations are performed, the user can select an entering variable by clicking on the variable name on the current tableau; the corresponding minimum ratio and the leaving variable values are displayed on the navigational form along with the change in the objective function. (See Figure CS14.7.) When the optimal solution is found, the navigational form changes again to display a "View Report" button. [See Figure CS14.6(b).] This button takes the user to the report sheet. We discuss the "Re-solve" button in Section CS24.5.



Using the Simplex Method to solve a Linear Programming problem.

Apply simplex method to your LP. The LP example must be given below in a specific format. To see an example of the format, click on **See Example**. After you are familiar with the format, either click on **Show Each Iteration** or **Show Final Solution**. In either case, you will be asked to specify the number of variables and the number of constraints, and whether you wish to maximize or minimize the objective function. Once you specify these, a simplex tableau of appropriate size will be constructed for you where you can enter the data in the required format.

The application will always select a nonbasic variable with the greatest violation as the entering variable. However, **you can select a different entering variable by selecting the variable you wish to enter.**

Initial Tableau

	z	x1	x2	x3	x4	x5	x6	x7	RHS	Ratio
z	1	-2	-1	-5	3	0	0	0	0	
x5	0	1	2	4	-1	1	0	0	6	1.5
x6	0	2	3	-1	1	0	1	0	12	-
x7	0	1	0	1	1	0	0	1	4	4.0

Iteration: 1

	z	x1	x2	x3	x4	x5	x6	x7	RHS	Ratio
z	1	-0.75	1.5	0	1.75	1.25	0	0	7.5	
x3	0	0.25	0.5	1	-0.25	0.25	0	0	1.5	6.0
x6	0	2.25	3.5	0	0.75	0.25	1	0	13.5	6.0
x7	0	0.75	-0.5	0	1.25	-0.25	0	1	2.5	3.3

The application automatically selects an entering variable. You may change the selection by clicking on a different nonbasic variable. Then, click on Show Next Iteration.

Entering variable: x1
Minimum ratio: 3.3
Leaving variable: x7
Change in the objective function: 2.475

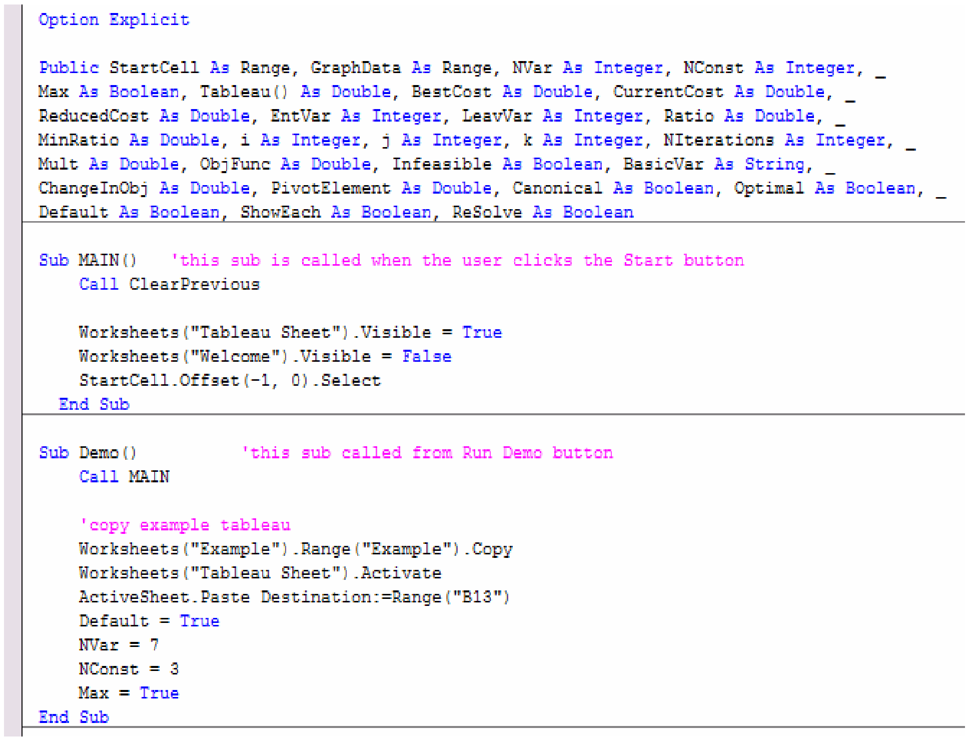
Figure CS14.7 The input sheet during the simplex method animation.

The "End" button on the input sheet and the report sheet takes the user back to the welcome sheet. The "See Example" button takes the user to the example sheet, and the "Return to Tableau" button on the example sheet and on the report sheet returns the user to the input sheet.

Summary	**Navigational buttons**	"Start" and "Run Demo" on the welcome sheet; "End" on the input sheet and report sheet; "See Example" on the input sheet; "Return to Tableau" on the example sheet and report sheet.
	Navigational form	A dynamic form with several user options; it is always displayed when the user is on the input sheet.
	Input form	Prompts the user for the problem parameters.

CS14.4 *Procedures*

We will now outline the procedures for this application beginning with the initial sub procedures and the variable definitions. (See Figure CS14.8.) The *Main* procedure, which is called from the “Start” button, calls the *ClearPrevious* procedure and takes the user to the input sheet. The *Demo* procedure calls the *Main* procedure and copies and pastes the demo initial tableau on the input sheet; it also initializes the problem parameters so the input form does not need to be shown.



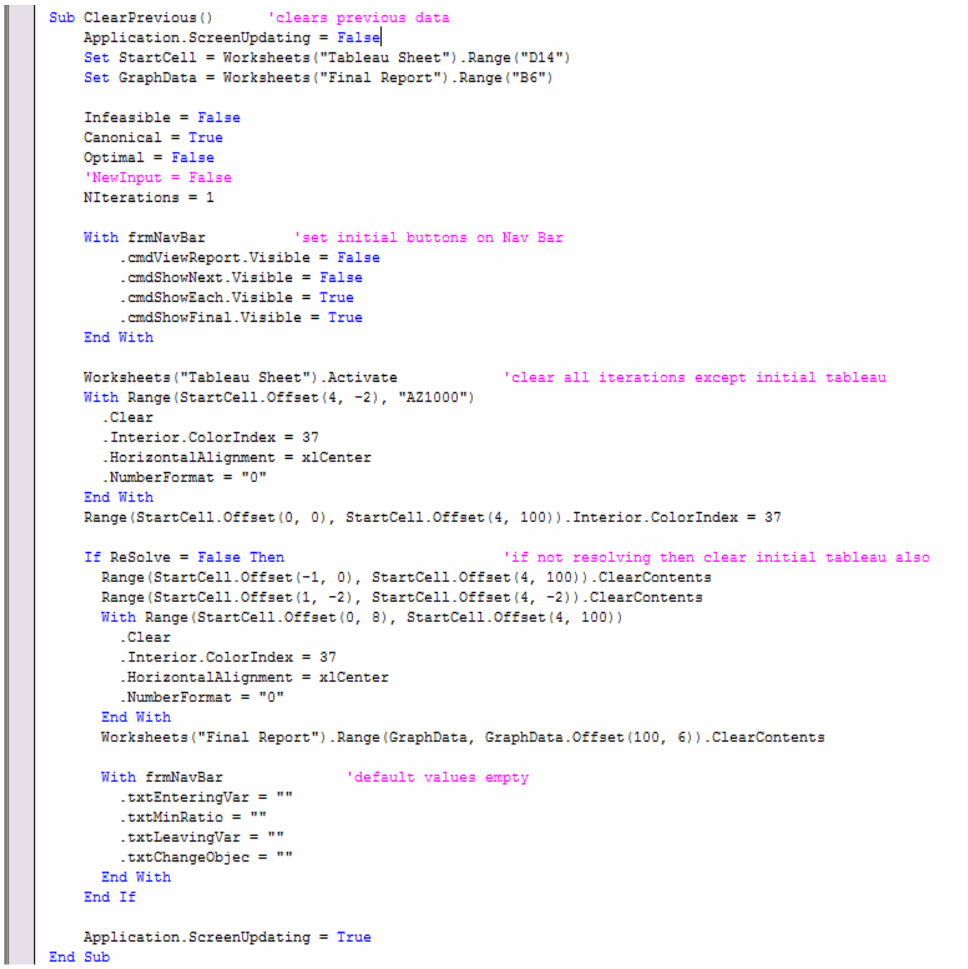
```
Option Explicit

Public StartCell As Range, GraphData As Range, NVar As Integer, NConst As Integer, _
Max As Boolean, Tableau() As Double, BestCost As Double, CurrentCost As Double, _
ReducedCost As Double, EntVar As Integer, LeavVar As Integer, Ratio As Double, _
MinRatio As Double, i As Integer, j As Integer, k As Integer, NIterations As Integer, _
Mult As Double, ObjFunc As Double, Infeasible As Boolean, BasicVar As String, _
ChangeInObj As Double, PivotElement As Double, Canonical As Boolean, Optimal As Boolean, _
Default As Boolean, ShowEach As Boolean, ReSolve As Boolean

Sub MAIN()   'this sub is called when the user clicks the Start button
    Call ClearPrevious

    Worksheets("Tableau Sheet").Visible = True
    Worksheets("Welcome").Visible = False
    StartCell.Offset(-1, 0).Select
  End Sub

Sub Demo()          'this sub called from Run Demo button
    Call MAIN

    'copy example tableau
    Worksheets("Example").Range("Example").Copy
    Worksheets("Tableau Sheet").Activate
    ActiveSheet.Paste Destination:=Range("B13")
    Default = True
    NVar = 7
    NConst = 3
    Max = True
End Sub
```

Figure CS14.8 The variable declarations and the *Main* and *Demo* procedures.

The *ClearPrevious* procedure clears any previous data and initializes several variables. (See Figure CS14.9.) It also initializes the input sheet formatting as well as the navigational form buttons and default values.

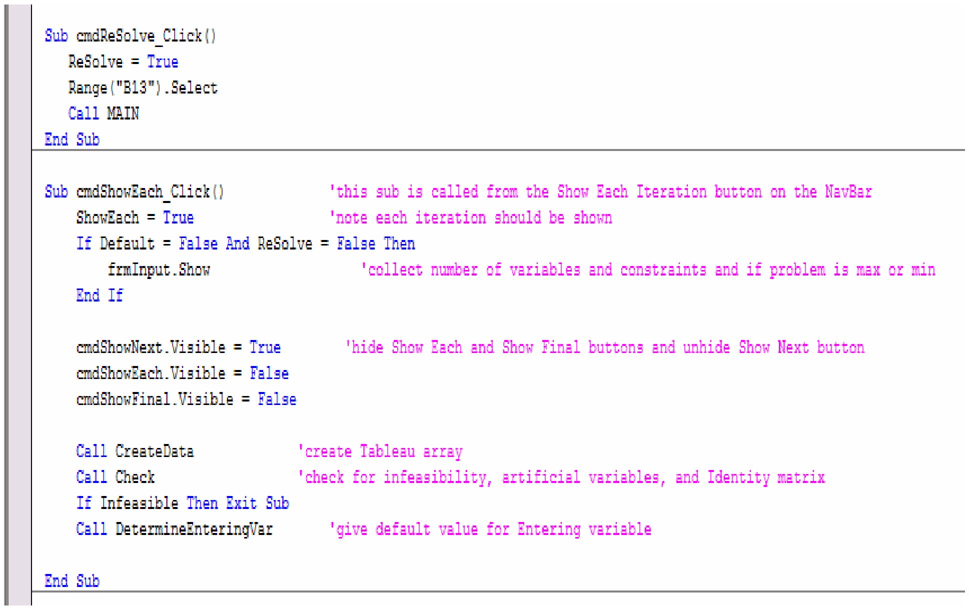
The navigational form has several procedures. (We will discuss the “Re-solve” button procedures in Section CS24.5.) The “Show Each Iteration” button procedure begins by displaying the input form. (See Figure CS14.10.) It then updates the buttons on the form and calls the *CreateData*, *Check*, and *DetermineEnteringVar* procedures.

The procedure for the “Show Final Solution” button also begins by showing the user the input form and updating the buttons on the form. (See Figure CS14.11.) It then also calls the *CreateData* and *Check* procedures. Next, it performs a loop in which the iterations occur until the solution is optimal. These iterations involve calling the *DetermineEnteringVar*, *DetermineLeavingVars*, *PerformPivotOps*, and *NewTableau* procedures. Once the optimal solution has been determined, the form buttons are again updated and a message box is displayed to the user.



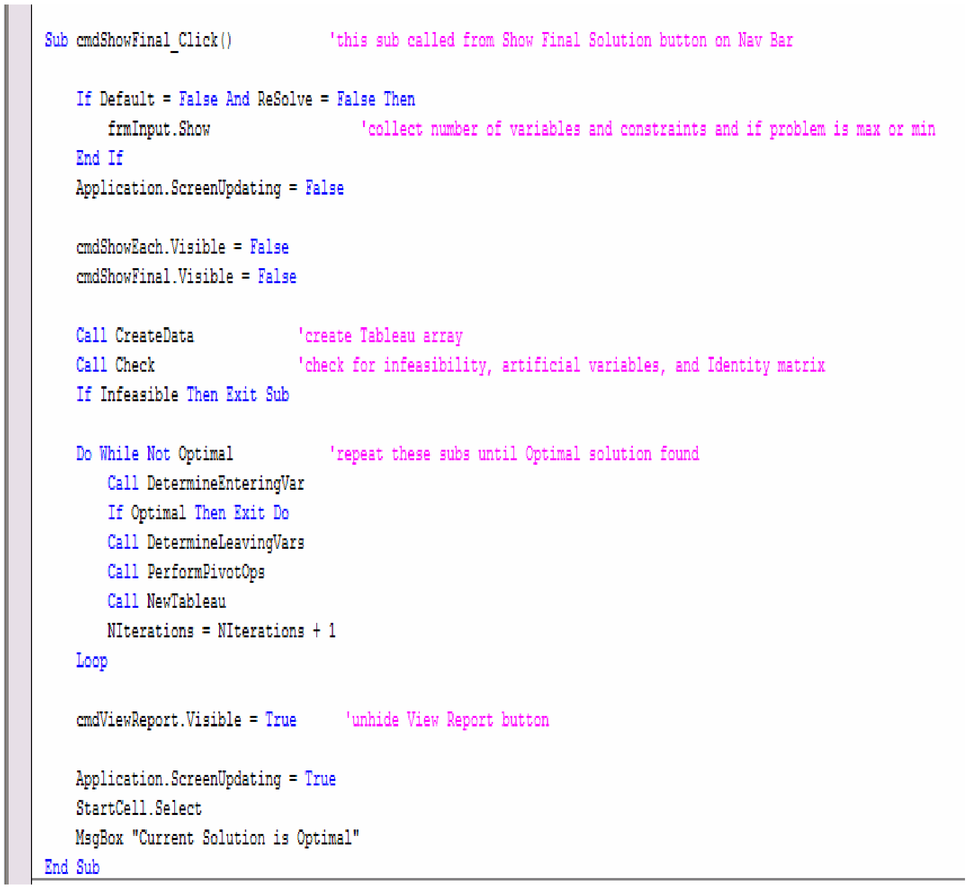
```
Sub ClearPrevious()      'clears previous data
    Application.ScreenUpdating = False
    Set StartCell = Worksheets("Tableau Sheet").Range("D14")
    Set GraphData = Worksheets("Final Report").Range("B6")

    Infeasible = False
    Canonical = True
    Optimal = False
    'NewInput = False
    NIterations = 1

    With frmNavBar          'set initial buttons on Nav Bar
        .cmdViewReport.Visible = False
        .cmdShowNext.Visible = False
        .cmdShowEach.Visible = True
        .cmdShowFinal.Visible = True
    End With

    Worksheets("Tableau Sheet").Activate            'clear all iterations except initial tableau
    With Range(StartCell.Offset(4, -2), "AZ1000")
      .Clear
      .Interior.ColorIndex = 37
      .HorizontalAlignment = xlCenter
      .NumberFormat = "0"
    End With
    Range(StartCell.Offset(0, 0), StartCell.Offset(4, 100)).Interior.ColorIndex = 37

    If ReSolve = False Then                          'if not resolving then clear initial tableau also
      Range(StartCell.Offset(-1, 0), StartCell.Offset(4, 100)).ClearContents
      Range(StartCell.Offset(1, -2), StartCell.Offset(4, -2)).ClearContents
      With Range(StartCell.Offset(0, 8), StartCell.Offset(4, 100))
        .Clear
        .Interior.ColorIndex = 37
        .HorizontalAlignment = xlCenter
        .NumberFormat = "0"
      End With
      Worksheets("Final Report").Range(GraphData, GraphData.Offset(100, 6)).ClearContents

      With frmNavBar              'default values empty
        .txtEnteringVar = ""
        .txtMinRatio = ""
        .txtLeavingVar = ""
        .txtChangeObjec = ""
      End With
    End If

    Application.ScreenUpdating = True
End Sub
```

Figure CS14.9 The *ClearPrevious* procedure.



```
Sub cmdReSolve_Click()
   ReSolve = True
   Range("B13").Select
   Call MAIN
End Sub

Sub cmdShowEach_Click()             'this sub is called from the Show Each Iteration button on the NavBar
    ShowEach = True                 'note each iteration should be shown
    If Default = False And ReSolve = False Then
        frmInput.Show                   'collect number of variables and constraints and if problem is max or min
    End If

    cmdShowNext.Visible = True        'hide Show Each and Show Final buttons and unhide Show Next button
    cmdShowEach.Visible = False
    cmdShowFinal.Visible = False

    Call CreateData             'create Tableau array
    Call Check                  'check for infeasibility, artificial variables, and Identity matrix
    If Infeasible Then Exit Sub
    Call DetermineEnteringVar       'give default value for Entering variable

End Sub
```

Figure CS14.10 The navigational form *cmdRe-solve_Click* and *cmdShowEach_Click* procedures.

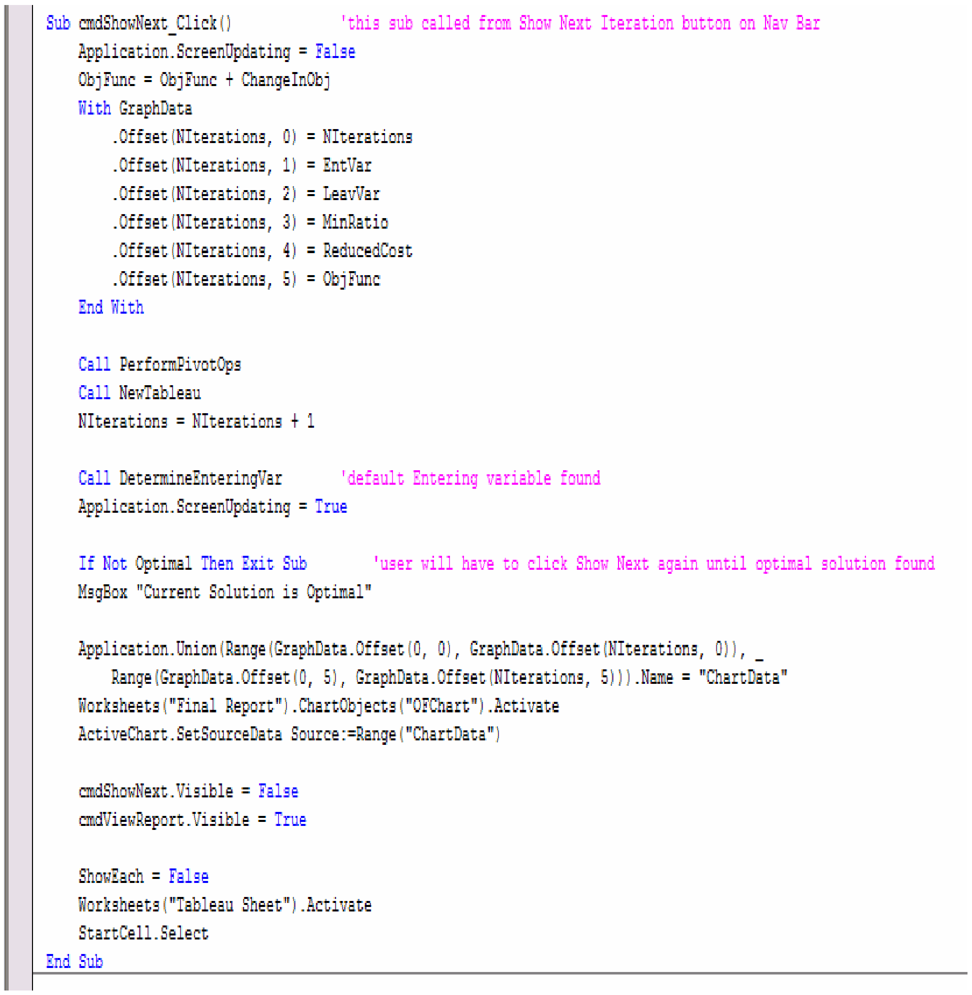


```
Sub cmdShowFinal_Click()          'this sub called from Show Final Solution button on Nav Bar

    If Default = False And ReSolve = False Then
        frmInput.Show                  'collect number of variables and constraints and if problem is max or min
    End If
    Application.ScreenUpdating = False

    cmdShowEach.Visible = False
    cmdShowFinal.Visible = False

    Call CreateData            'create Tableau array
    Call Check                 'check for infeasibility, artificial variables, and Identity matrix
    If Infeasible Then Exit Sub

    Do While Not Optimal          'repeat these subs until Optimal solution found
        Call DetermineEnteringVar
        If Optimal Then Exit Do
        Call DetermineLeavingVars
        Call PerformPivotOps
        Call NewTableau
        NIterations = NIterations + 1
    Loop

    cmdViewReport.Visible = True      'unhide View Report button

    Application.ScreenUpdating = True
    StartCell.Select
    MsgBox "Current Solution is Optimal"
End Sub
```

Figure CS14.11 The navigational form *cmdShowFinal_Click* procedure.

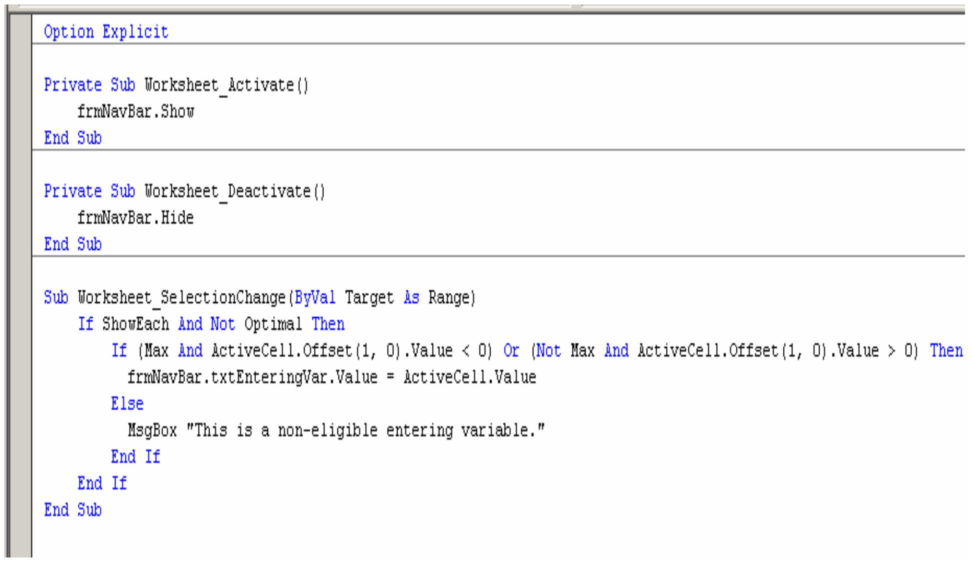
If the user initially selected the "Show Each Iteration" button, then the procedure for the "Show Next Iteration" button is used. (See Figure CS14.12.) Note that this procedure is called after the user selects the entering variable and a corresponding leaving variable has been determined. This procedure therefore begins by updating the objective function value and reporting the current iteration values on the report sheet's summary table. It then calls the *PerformPivotOps* and *NewTableau* procedures to progress to the next iteration. The *DetermineEnteringVar* procedure selects a default entering variable, and if no entering variables are found, then the solution is optimal and the report sheet is updated.

As the user selects an entering variable on the input sheet, the *Worksheet_SelectionChange* procedure displays it on the navigational form. (See Figure CS14.13.) As this form text box is changed, a procedure for the navigational form is called; it calls the *DetermineLeavingVars* procedure and updates the minimum ratio and leaving variable text boxes on the form. (See Figure CS14.14.) The input sheet and navigational form also have procedures to ensure that the navigational form is always displayed when the input sheet is active and hidden if the form is closed. (See Figures CS24.13 and CS24.14.) The procedure for the "View Report" button simply takes the user to the report sheet. (See Figure CS14.14.)



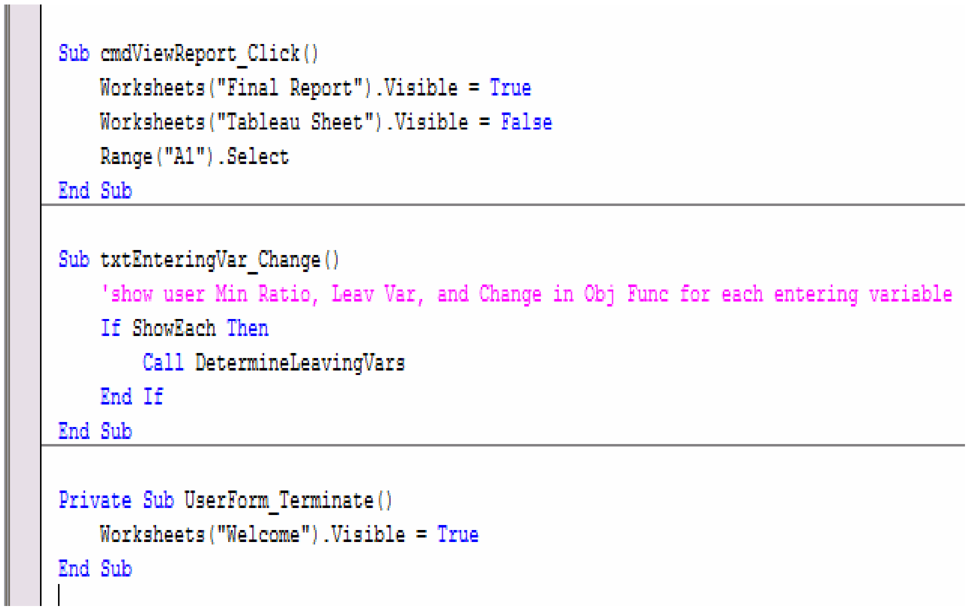
```
Sub cmdShowNext_Click()            'this sub called from Show Next Iteration button on Nav Bar
    Application.ScreenUpdating = False
    ObjFunc = ObjFunc + ChangeInObj
    With GraphData
        .Offset(NIterations, 0) = NIterations
        .Offset(NIterations, 1) = EntVar
        .Offset(NIterations, 2) = LeavVar
        .Offset(NIterations, 3) = MinRatio
        .Offset(NIterations, 4) = ReducedCost
        .Offset(NIterations, 5) = ObjFunc
    End With

    Call PerformPivotOps
    Call NewTableau
    NIterations = NIterations + 1

    Call DetermineEnteringVar       'default Entering variable found
    Application.ScreenUpdating = True

    If Not Optimal Then Exit Sub        'user will have to click Show Next again until optimal solution found
    MsgBox "Current Solution is Optimal"

    Application.Union(Range(GraphData.Offset(0, 0), GraphData.Offset(NIterations, 0)), _
        Range(GraphData.Offset(0, 5), GraphData.Offset(NIterations, 5))).Name = "ChartData"
    Worksheets("Final Report").ChartObjects("OFChart").Activate
    ActiveChart.SetSourceData Source:=Range("ChartData")

    cmdShowNext.Visible = False
    cmdViewReport.Visible = True

    ShowEach = False
    Worksheets("Tableau Sheet").Activate
    StartCell.Select
End Sub
```

Figure CS14.12 The navigational form *cmdShowNext_Click* procedure.



```
Option Explicit

Private Sub Worksheet_Activate()
    frmNavBar.Show
End Sub

Private Sub Worksheet_Deactivate()
    frmNavBar.Hide
End Sub

Sub Worksheet_SelectionChange(ByVal Target As Range)
    If ShowEach And Not Optimal Then
        If (Max And ActiveCell.Offset(1, 0).Value < 0) Or (Not Max And ActiveCell.Offset(1, 0).Value > 0) Then
          frmNavBar.txtEnteringVar.Value = ActiveCell.Value
        Else
          MsgBox "This is a non-eligible entering variable."
        End If
    End If
End Sub
```

Figure CS14.13 Procedures for the input sheet.



```
Sub cmdViewReport_Click()
    Worksheets("Final Report").Visible = True
    Worksheets("Tableau Sheet").Visible = False
    Range("A1").Select
End Sub

Sub txtEnteringVar_Change()
    'show user Min Ratio, Leav Var, and Change in Obj Func for each entering variable
    If ShowEach Then
        Call DetermineLeavingVars
    End If
End Sub

Private Sub UserForm_Terminate()
    Worksheets("Welcome").Visible = True
End Sub
```

Figure CS14.14 The navigational form cmdViewReport_Click, UserForm_Terminate, and txtEnteringVar_Change procedures.

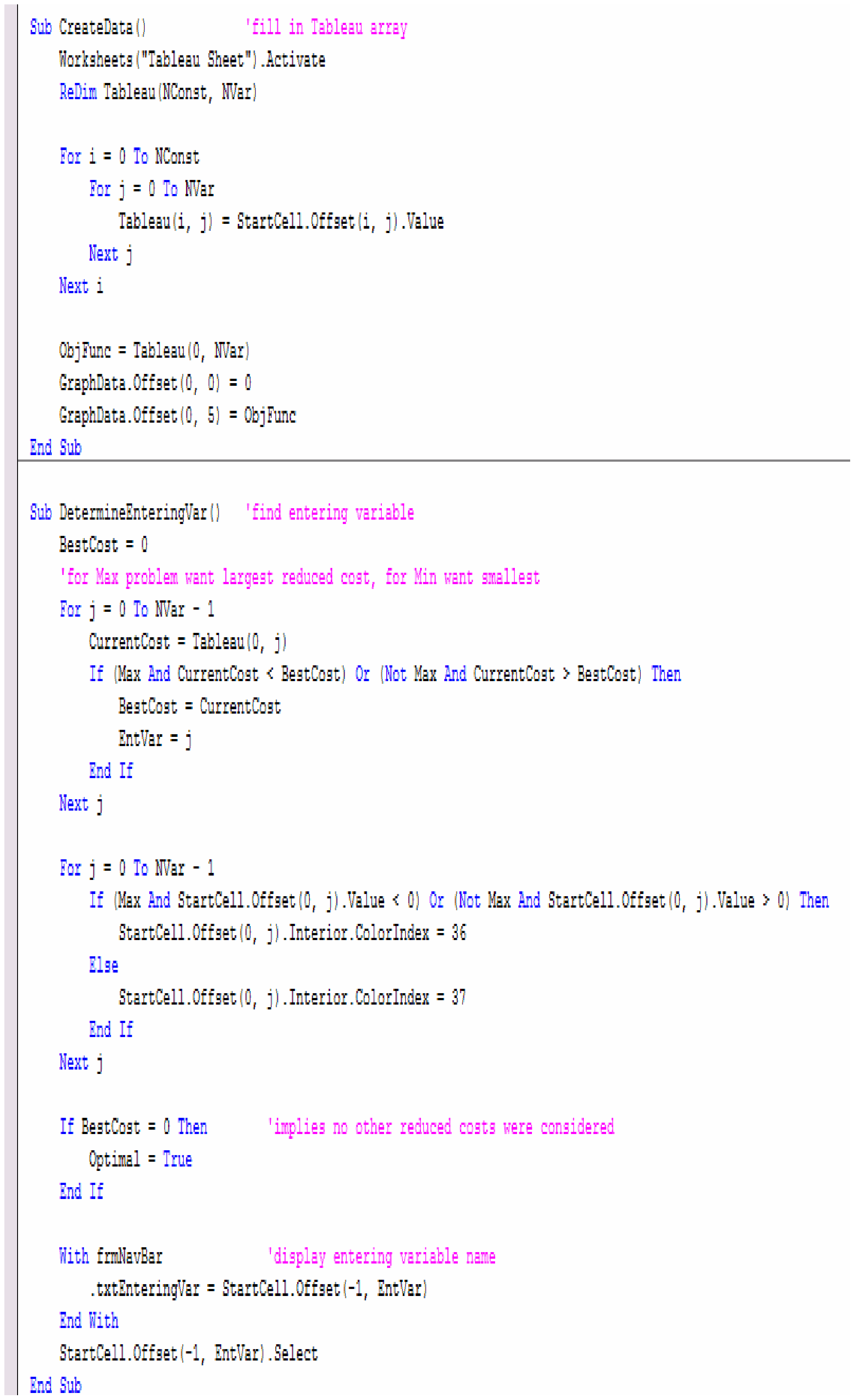
The *CreateData* procedure is initially called to populate an array with the values provided by the user in the initial tableau. (See Figure CS14.15.) It is then called for each iteration to update this array with the new tableau values. The *DetermineEnteringVar* procedure finds the default entering variable by scanning the objective function coefficients. (See Figure CS14.15.) If no coefficient is found that meets the entering variable criteria for the maximization or minimization problem, then the solution is considered to be optimal.

The *DetermineLeavingVars* procedure finds the leaving variable for the selected entering variable by determining the minimum ratio. (See Figure CS14.16.) If no minimum ratio is found, then the user is notified that the problem is unbounded. Otherwise, the change in the objective function is determined, and this value along with the minimum ratio and found leaving variable are displayed in the navigational form.

The *PerformPivotOps* procedure performs the pivot operations to switch the entering variable and leaving variable in the tableau. (See Figure CS14.17.)

The *NewTableau* procedure creates the new tableau for the next iteration. (See Figure CS14.18.) It enters the new values found after the *PerformPivotOps* procedure is completed.

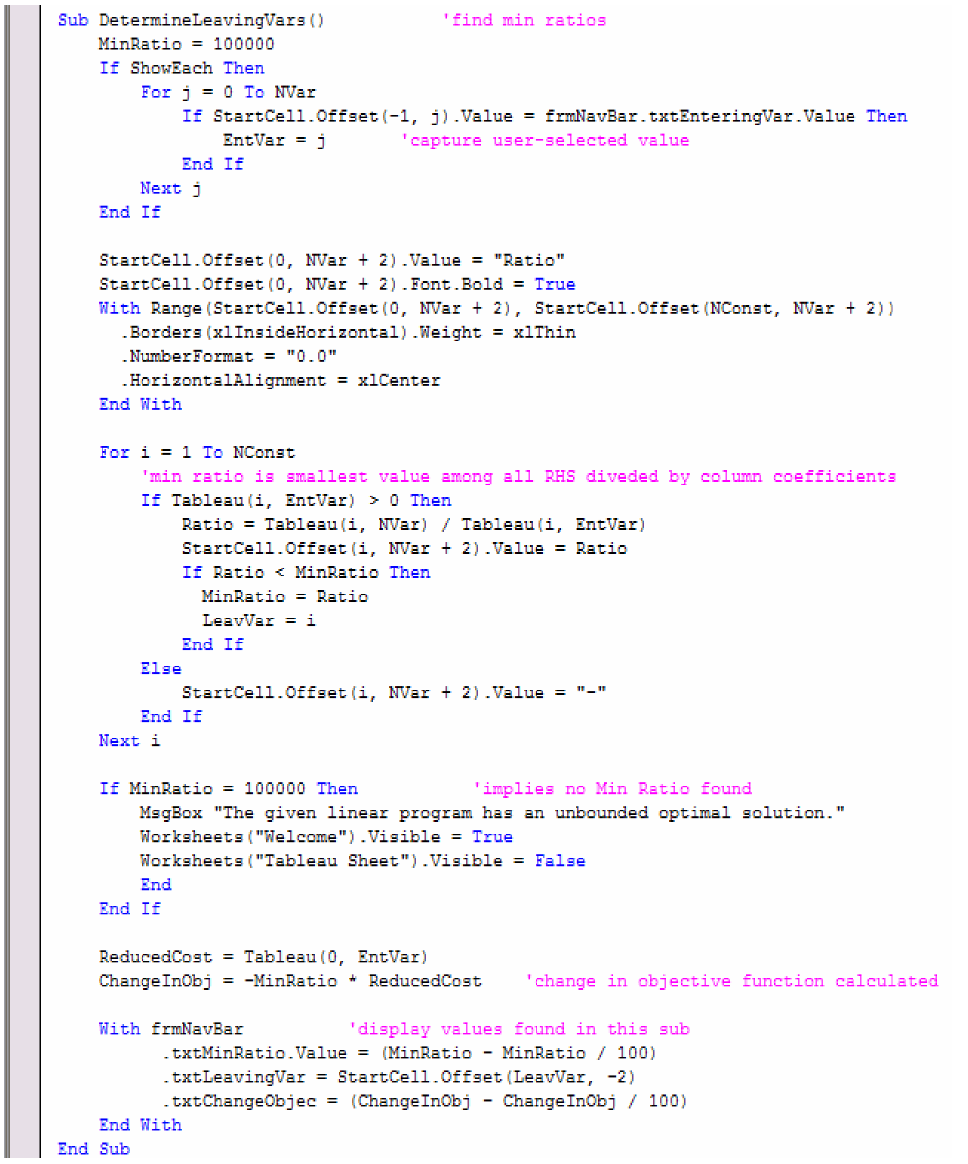
The *Check* procedure checks the feasibility of the initial tableau entered by the user. (See Figure CS14.19.) It ensures that the RHS values are non-negative, that any artificial variables are the initial basic variables, and that these artificial variables form an identity matrix in the constraint coefficients. If the tableau does not reflect a canonical problem format, then the tableau is transformed for the user.



```
Sub CreateData()            'fill in Tableau array
    Worksheets("Tableau Sheet").Activate
    ReDim Tableau(NConst, NVar)

    For i = 0 To NConst
        For j = 0 To NVar
            Tableau(i, j) = StartCell.Offset(i, j).Value
        Next j
    Next i

    ObjFunc = Tableau(0, NVar)
    GraphData.Offset(0, 0) = 0
    GraphData.Offset(0, 5) = ObjFunc
End Sub
```

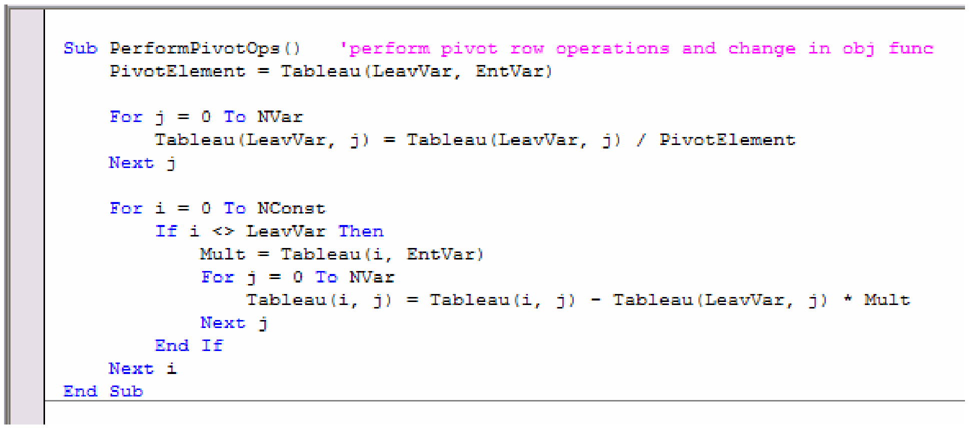
```
Sub DetermineEnteringVar()   'find entering variable
    BestCost = 0
    'for Max problem want largest reduced cost, for Min want smallest
    For j = 0 To NVar - 1
        CurrentCost = Tableau(0, j)
        If (Max And CurrentCost < BestCost) Or (Not Max And CurrentCost > BestCost) Then
            BestCost = CurrentCost
            EntVar = j
        End If
    Next j

    For j = 0 To NVar - 1
        If (Max And StartCell.Offset(0, j).Value < 0) Or (Not Max And StartCell.Offset(0, j).Value > 0) Then
            StartCell.Offset(0, j).Interior.ColorIndex = 36
        Else
            StartCell.Offset(0, j).Interior.ColorIndex = 37
        End If
    Next j

    If BestCost = 0 Then        'implies no other reduced costs were considered
        Optimal = True
    End If

    With frmNavBar              'display entering variable name
        .txtEnteringVar = StartCell.Offset(-1, EntVar)
    End With
    StartCell.Offset(-1, EntVar).Select
End Sub
```

Figure CS14.15 The CreateData and DetermineEnteringVar procedures.



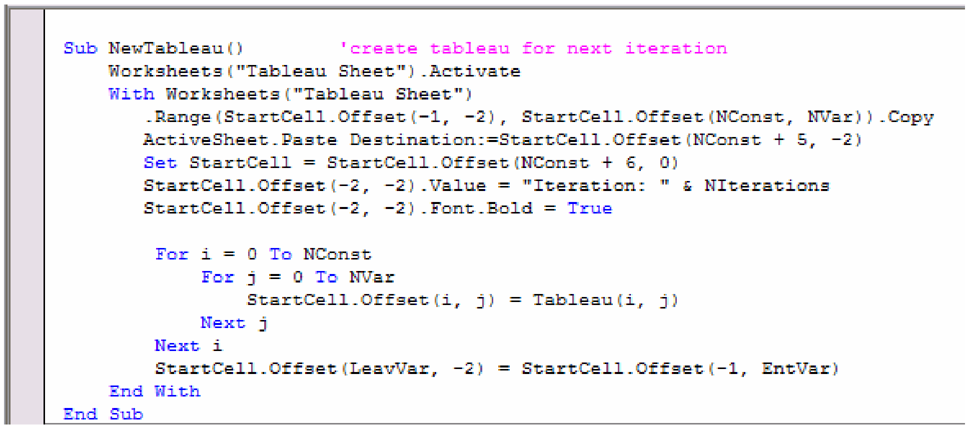
```
Sub DetermineLeavingVars()          'find min ratios
    MinRatio = 100000
    If ShowEach Then
        For j = 0 To NVar
            If StartCell.Offset(-1, j).Value = frmNavBar.txtEnteringVar.Value Then
                EntVar = j       'capture user-selected value
            End If
        Next j
    End If

    StartCell.Offset(0, NVar + 2).Value = "Ratio"
    StartCell.Offset(0, NVar + 2).Font.Bold = True
    With Range(StartCell.Offset(0, NVar + 2), StartCell.Offset(NConst, NVar + 2))
      .Borders(xlInsideHorizontal).Weight = xlThin
      .NumberFormat = "0.0"
      .HorizontalAlignment = xlCenter
    End With

    For i = 1 To NConst
        'min ratio is smallest value among all RHS diveded by column coefficients
        If Tableau(i, EntVar) > 0 Then
            Ratio = Tableau(i, NVar) / Tableau(i, EntVar)
            StartCell.Offset(i, NVar + 2).Value = Ratio
            If Ratio < MinRatio Then
              MinRatio = Ratio
              LeavVar = i
            End If
        Else
            StartCell.Offset(i, NVar + 2).Value = "-"
        End If
    Next i

    If MinRatio = 100000 Then           'implies no Min Ratio found
        MsgBox "The given linear program has an unbounded optimal solution."
        Worksheets("Welcome").Visible = True
        Worksheets("Tableau Sheet").Visible = False
        End
    End If

    ReducedCost = Tableau(0, EntVar)
    ChangeInObj = -MinRatio * ReducedCost    'change in objective function calculated

    With frmNavBar          'display values found in this sub
          .txtMinRatio.Value = (MinRatio - MinRatio / 100)
          .txtLeavingVar = StartCell.Offset(LeavVar, -2)
          .txtChangeObjec = (ChangeInObj - ChangeInObj / 100)
    End With
End Sub
```

Figure CS14.16 The DetermineLeavingVars procedure.



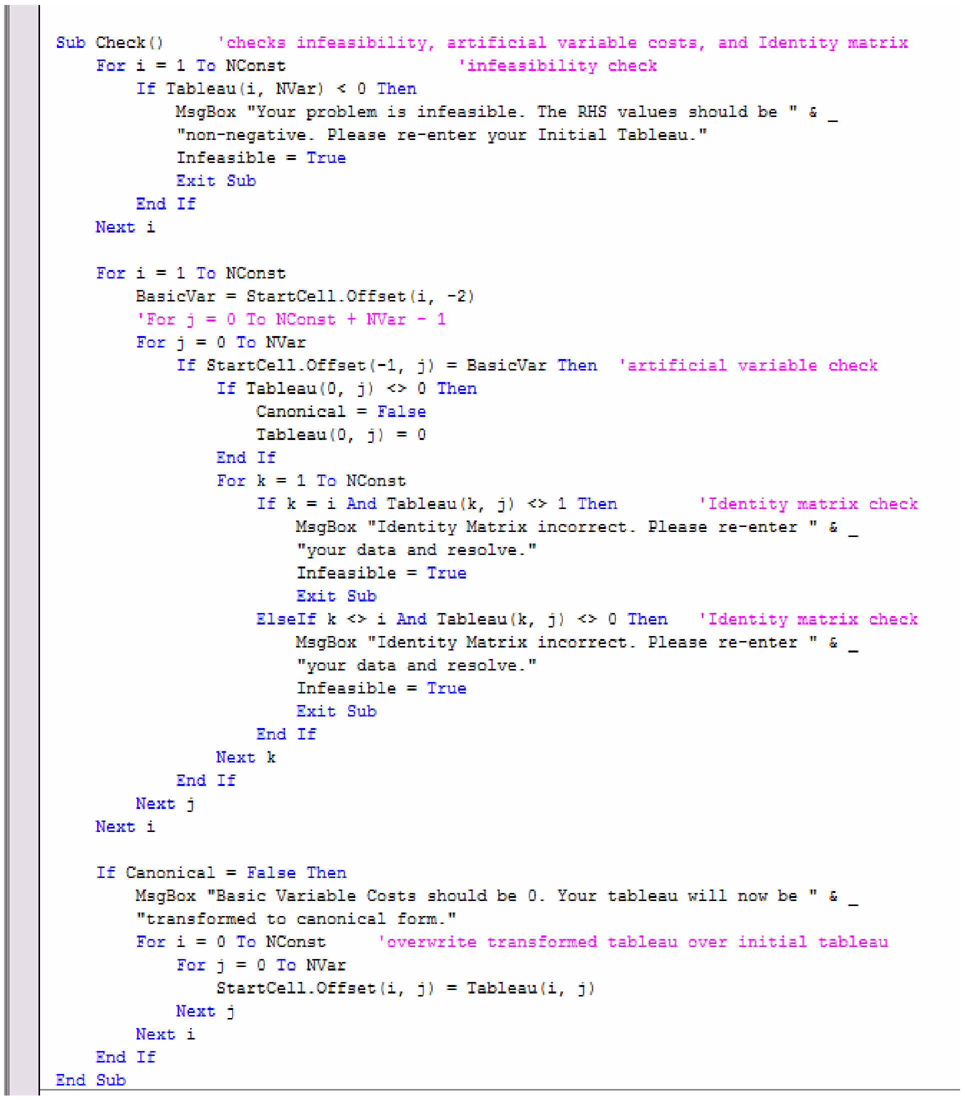
```
Sub PerformPivotOps()   'perform pivot row operations and change in obj func
    PivotElement = Tableau(LeavVar, EntVar)

    For j = 0 To NVar
        Tableau(LeavVar, j) = Tableau(LeavVar, j) / PivotElement
    Next j

    For i = 0 To NConst
        If i <> LeavVar Then
            Mult = Tableau(i, EntVar)
            For j = 0 To NVar
                Tableau(i, j) = Tableau(i, j) - Tableau(LeavVar, j) * Mult
            Next j
        End If
    Next i
End Sub
```

Figure CS14.17 The *PerformPivotOps* procedure.



```
Sub NewTableau()          'create tableau for next iteration
    Worksheets("Tableau Sheet").Activate
    With Worksheets("Tableau Sheet")
       .Range(StartCell.Offset(-1, -2), StartCell.Offset(NConst, NVar)).Copy
       ActiveSheet.Paste Destination:=StartCell.Offset(NConst + 5, -2)
       Set StartCell = StartCell.Offset(NConst + 6, 0)
       StartCell.Offset(-2, -2).Value = "Iteration: " & NIterations
       StartCell.Offset(-2, -2).Font.Bold = True

        For i = 0 To NConst
            For j = 0 To NVar
                StartCell.Offset(i, j) = Tableau(i, j)
            Next j
        Next i
        StartCell.Offset(LeavVar, -2) = StartCell.Offset(-1, EntVar)
    End With
End Sub
```

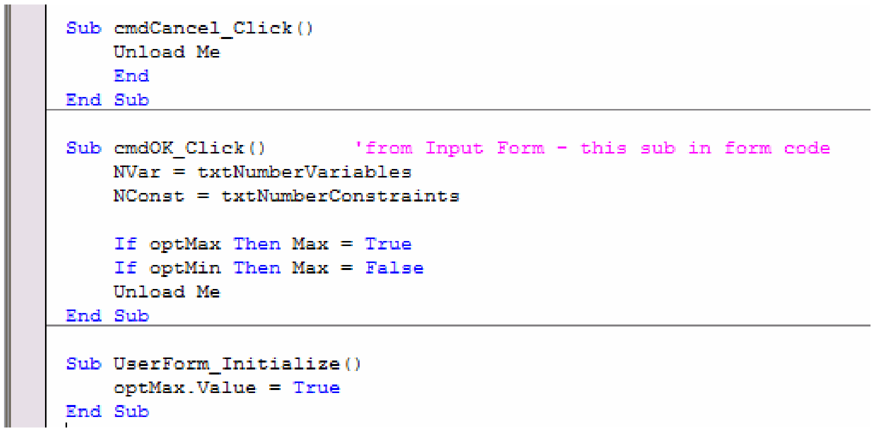
Figure CS14.18 The *NewTableau* procedure.



```
Sub Check()      'checks infeasibility, artificial variable costs, and Identity matrix
    For i = 1 To NConst                    'infeasibility check
        If Tableau(i, NVar) < 0 Then
            MsgBox "Your problem is infeasible. The RHS values should be " & _
            "non-negative. Please re-enter your Initial Tableau."
            Infeasible = True
            Exit Sub
        End If
    Next i

    For i = 1 To NConst
        BasicVar = StartCell.Offset(i, -2)
        'For j = 0 To NConst + NVar - 1
        For j = 0 To NVar
            If StartCell.Offset(-1, j) = BasicVar Then  'artificial variable check
                If Tableau(0, j) <> 0 Then
                    Canonical = False
                    Tableau(0, j) = 0
                End If
                For k = 1 To NConst
                    If k = i And Tableau(k, j) <> 1 Then        'Identity matrix check
                        MsgBox "Identity Matrix incorrect. Please re-enter " & _
                        "your data and resolve."
                        Infeasible = True
                        Exit Sub
                    ElseIf k <> i And Tableau(k, j) <> 0 Then   'Identity matrix check
                        MsgBox "Identity Matrix incorrect. Please re-enter " & _
                        "your data and resolve."
                        Infeasible = True
                        Exit Sub
                    End If
                Next k
            End If
        Next j
    Next i

    If Canonical = False Then
        MsgBox "Basic Variable Costs should be 0. Your tableau will now be " & _
        "transformed to canonical form."
        For i = 0 To NConst      'overwrite transformed tableau over initial tableau
            For j = 0 To NVar
                StartCell.Offset(i, j) = Tableau(i, j)
            Next j
        Next i
    End If
End Sub
```

Figure CS14.19 The *Check* procedure.

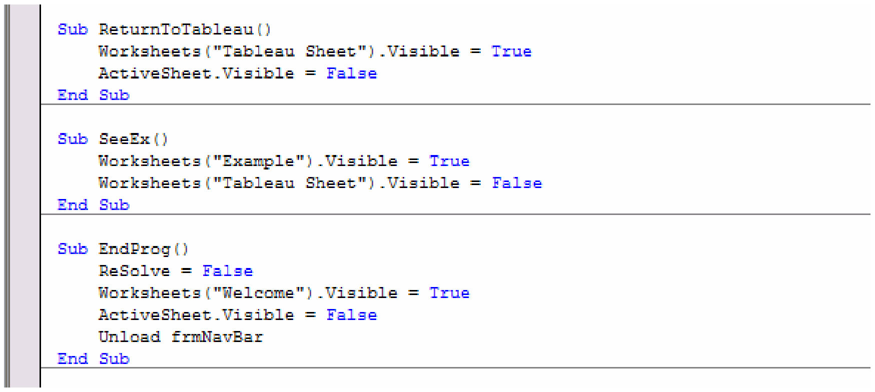
The procedures for the input form simply record the number of variables and constraints and the type of objective function. (See Figure CS14.20.)



```
Sub cmdCancel_Click()
    Unload Me
    End
End Sub

Sub cmdOK_Click()       'from Input Form - this sub in form code
    NVar = txtNumberVariables
    NConst = txtNumberConstraints

    If optMax Then Max = True
    If optMin Then Max = False
    Unload Me
End Sub

Sub UserForm_Initialize()
    optMax.Value = True
End Sub
```

Figure CS14.20 The input form procedures.

The navigational procedures are for the “Return to Tableau” buttons, the “See Example” button, and the “End” buttons. (See Figure CS14.21.)



```
Sub ReturnToTableau()
    Worksheets("Tableau Sheet").Visible = True
    ActiveSheet.Visible = False
End Sub

Sub SeeEx()
    Worksheets("Example").Visible = True
    Worksheets("Tableau Sheet").Visible = False
End Sub

Sub EndProg()
    ReSolve = False
    Worksheets("Welcome").Visible = True
    ActiveSheet.Visible = False
    Unload frmNavBar
End Sub
```

Figure CS14.21 The navigational procedures.

Summary

Main	Initializes application and takes user to input sheet.
RunDemo	Takes user to input sheet and copies initial tableau from example sheet.
ClearPrevious	Clears previous tableau values and initializes variables, formatting, and default values.
Navigational form procedures	Procedures for “Show Each Iteration”, “Show Final Solution”, “Show Next Iteration”, “View Report”, and “Re-solve” buttons.
Input form procedures	Records problem parameters.
CreateData	Stores tableau values.
Check	Checks if initial tableau is feasible and in canonical form.
DetermineEnteringVar	Finds default entering variable and determines if current solution is optimal.
DetermineLeavingVars	Finds minimum ratio, leaving variable, and change in objective function for selected entering variable.
PerformPivotOps	Performs pivot operations for current entering and leaving variables.
NewTableau	Creates tableau for next iteration with updated values.
Input sheet procedures	Records entering variable selected by user.
Navigational procedures	For “Return to Tableau”, “See Example”, and “End” buttons.

CS14.5 *Re-solve Options*

To re-solve this application, the user can press the navigational form's "Re-solve" button, which clears all tableaus except for the initial one. The user can then change some or all of the values in the initial tableau and re-solve the problem. The user can also change the number of variables or constraints in the tableau; the input form is redisplayed so he or she can enter these new parameter values and change the objective.

Summary	**"Re-solve"**	Clears all the iterations and allows the user to modify the initial tableau and re-solve the problem.

CS14.6 *Summary*

- This application animates the simplex method to solve a user-defined linear programming problem.
- This application requires four worksheets: the welcome sheet, the input sheet, the example sheet, and the report sheet.
- We use navigational buttons and two user forms for the user interface.
- Several procedures in this application initialize and perform the simplex method iterations.
- The user can re-solve the application by pressing the "Re-solve" button on the navigational form to clear all iterations and modify the initial tableau.

CS14.7 *Extensions*

- Allow the user to solve this problem with the dual simplex method.
- Perform sensitivity analysis for the user.
- Create a graph of the user's problem and animate the selection of all the extreme points as the iterations are performed.