CASE STUDY

eighteen

Sales Force Allocation

case study

OVERVIEW

CS18.1 *Application Overview and Model Development*

This application seeks to determine the optimal allocation of sales force in order to maximize profits. In this application, we consider a customer-call scenario in which a manager is trying to determine how many calls should be made to each customer in the upcoming period. We employ optimization to solve the problem, and a trade off curve option allows users to observe optimized results for various upper bound values on the total number of calls to make.

CS18.1.1 Model Definition and Assumptions

We begin by prompting the user to enter historical data about the sales force allocation. We assume that this historical data is known and includes the number of calls, unit sales, revenue per sale, and cost per call for all customers. This historical data is used to find s-curve parameters to estimate future profit. In marketing research, the s-curve is considered a more accurate model than the power curve. It is modeled as follows:

$$R = a + \frac{(b-a)S^c}{d+S^c}$$

The Solver finds for each customer the optimal s-curve parameters that minimize the sum of squared errors in the estimated profit and the calculated profit. Given the historical number of calls made to a customer, a variation of this value is found for 0.10, 0.50, 1.50, and 10 times the historical value. Aside from the historical number of calls, which has a corresponding historical unit sales value, the user is asked to estimate the unit sales for the variations of the number of calls. From each of these estimates, an estimated profit value is found. A calculated profit value is then determined using the s-curve parameters and the s-curve model provided above. The Solver is then used to find the best s-curve parameters that minimize the sum of squared errors between these two profit values.

Number of Calls	Unit Sales	Estimated Profit
=0.1* Historical NumberCalls	**(User estimate)**	=(CustRev*UnitSales-CustCost*NumberCalls)/1000
=0.5* Historical NumberCalls	**(User estimate)**	…
(Historical NumberCalls)	(Historical UnitSales)	
=1.5* Historical NumberCalls	**(User estimate)**	
=10* Historical NumberCalls	**(User estimate)**	

Calculated Profit	Error
=(Aparam+((Bparam-Aparam)*NumberCalls^Cparam) / (Dparam+ NumberCalls ^Cparam)) / 1000	=(Estimated - Calculated)^2
…	…
Minimize SSE	=SUM(Errors)

Parameters	
a	(decision variables)
b	
c	
d	

Once these s-curve parameters have been determined for each customer, they are used to calculate the profit in the main optimization model. This model seeks to find the number of calls that should be allocated to each customer so that the total profit over all the customers is maximized. We consider that there is a lower bound and an upper bound constraint for the number of calls that can be made to each customer. We also consider that there is an upper bound on the total number of calls that can be made over all the customers. For the trade-off curve option, we allow the user to vary this overall upper bound and view the change in the optimized total profit.

Customer	Lower Bound	Number of Calls	Upper Bound	Unit Sales
1	(Given by User)	(decision variables)	(Given by User)	=INT(Aparam + ((Baram – Aparam) * NumCalls^Cparam) / (Dparam+NumCalls^Cparam))
2				
...				
		=SUM(CallsDV)		

Profit / Sale	Cost / Call	Total Profit
(From Historical Data)	(From Historical Data)	=Profit/Sale*UnitSales – Cost/Call*NumberCalls

Maximize: =SUM(Profit)

See *Practical Management Science* by Winston and Albright for more details.

CS18.1.2 Input

The user is prompted for various inputs throughout this application:

- Historical input = number of calls, unit sales, revenue per sale, and cost per call for each customer
- Estimate input = expected unit sales for variations on the historical number of calls
- Optimization input = lower and upper bounds on the number of calls that can be made to each customer, upper bound on the total number of calls that can be made across all customers

CS18.1.3 Output

There are two main outputs for this application:

- Number of calls to make to each customer in order to maximize profits
- Trade-off curve of upper bound on total number of calls versus the total overall revenue

CS18.2 *Worksheets*

This application requires five worksheets: the welcome sheet, the historical data sheet, the s-curve parameters sheet, the optimization sheet, and the trade-off curve sheet. The welcome sheet contains the title, the description of the application, and the "Run Demo" and "Start" buttons. (See Figure CS18.1.) The "Run Demo" button copies some demo historical data and then takes the user to the historical data sheet. The "Start" button on the welcome sheet prompts the user for some input and then displays the historical data sheet.

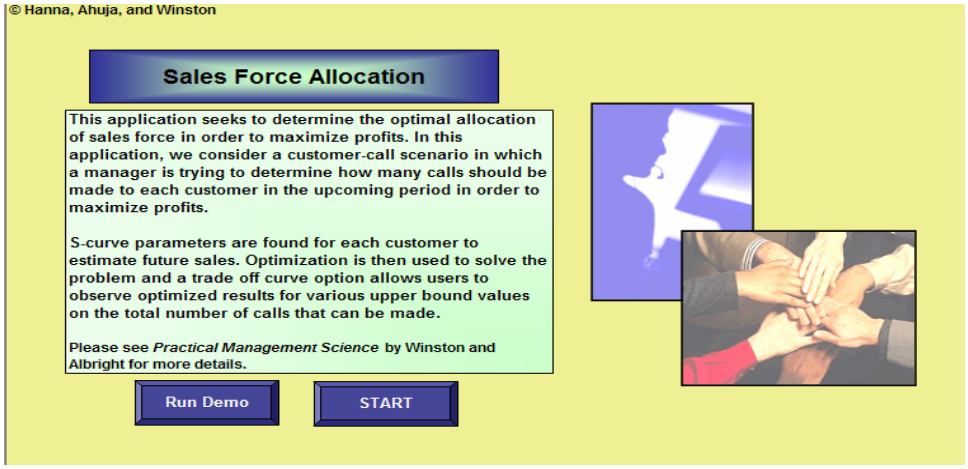


Figure CS18.1 The welcome sheet.

The historical data sheet stores the input for the historical number of calls made to each customer, the unit sales achieved, the profit per unit sale, and the cost per call made. The total profit is calculated per customer with a simple formula of unit sales * profit per sale – number of calls * cost per call. The total profit overall customers is also calculated as the sum of the total profit per customer. Figure CS18.2 presents the historical data sheet with the demo data.

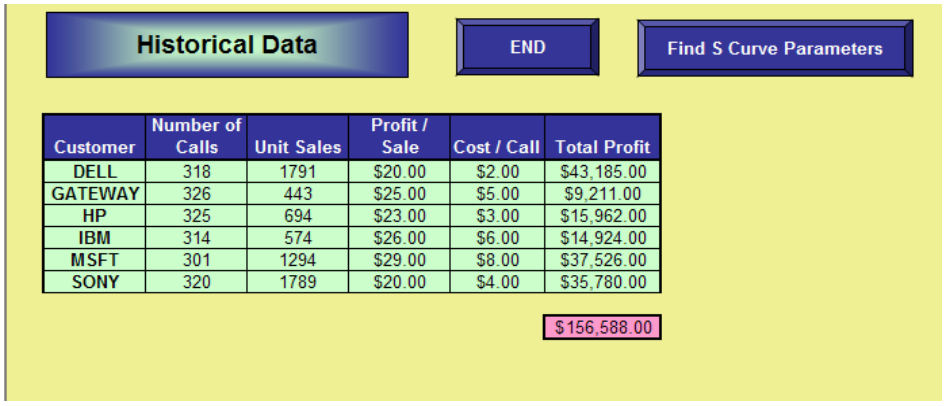


Figure CS18.2 The historical data sheet.

The "Find S Curve Parameters" button takes the user to the s-curve parameters sheet, which is where the user optimizes the s-curve parameters for each customer. (See Figure CS18.3.) For each customer, one at a time, the historical data is referenced as input on this sheet. The number of calls and unit sales in the historical data are found in the middle row of the main calculation table of this sheet. Using the historical number of calls, the

application calculates variations of .10, .50, 1.5, and 10 times the historical value. The user is then asked to estimate the unit sales for these variations in the number of calls. With the user's values, the application calculates an estimated profit for each scenario. The Solver then finds the a, b, c, and d parameter values for the s-curve such that the mean squared error between the calculated profit from these parameters and the estimated profit from the user's input are minimized. A chart comparing the estimated and calculated profits is shown.

Next, the user clicks the "Calculate" button to optimize these s-curve parameters for each customer after he or she has entered the sales estimates. When all the customer's parameters have been optimized, the user can press the "Go to Optimization" button to proceed to the optimization sheet.

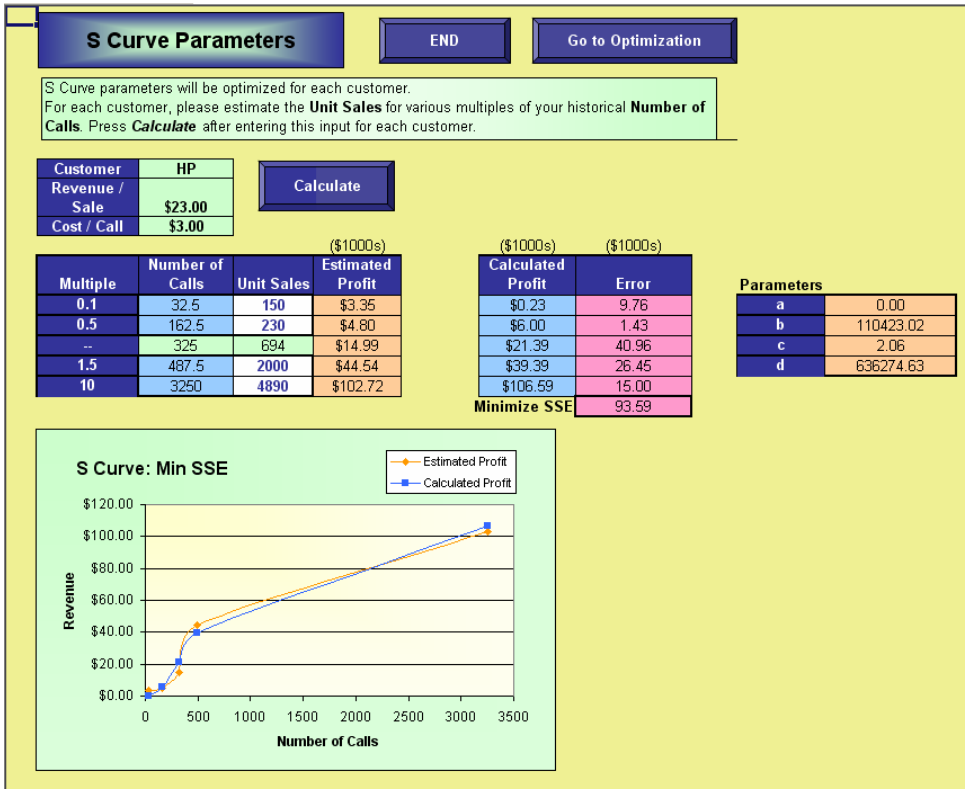
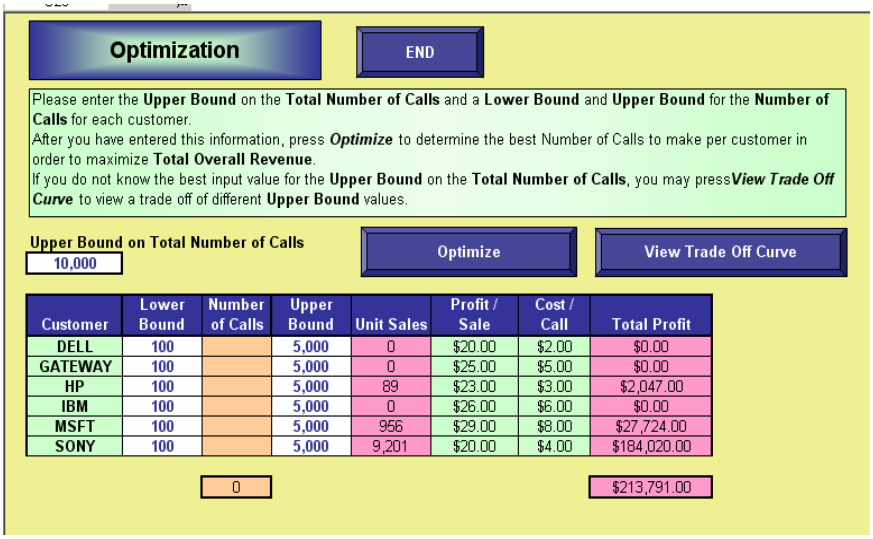


Figure CS18.3 The s-curve parameters sheet.

The main model of the application is solved on the optimization sheet. (See Figure CS18.4.) The user provides a lower bound and an upper bound on the number of calls that can be made to each customer; these are the decision variables. Using the optimized s-curve parameters and these decision variables, the unit sales are calculated. From these unit sales, the profit per sale, and the cost per call input, the total profit per customer is also calculated. The objective function is to maximize the sum of these total profit values for all the customers. When the user presses the "Optimize" button, the model is solved.

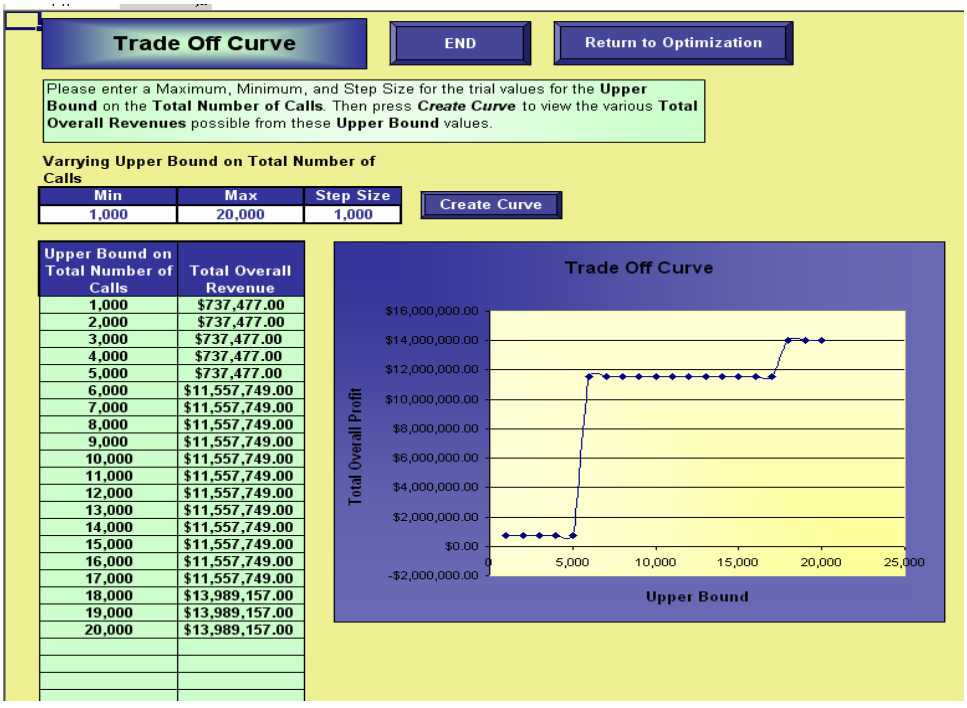
The user can also specify an upper bound constraint on the total number of calls for all the customers. By selecting the "View Trade Off Curve" button, the user can view the results of the different values of this upper bound for the total number of calls, which are listed on the trade-off curve sheet.



Customer	Lower Bound	Number of Calls	Upper Bound	Unit Sales	Profit / Sale	Cost / Call	Total Profit
DELL	100		5,000	0	$20.00	$2.00	$0.00
GATEWAY	100		5,000	0	$25.00	$5.00	$0.00
HP	100		5,000	89	$23.00	$3.00	$2,047.00
IBM	100		5,000	0	$26.00	$6.00	$0.00
MSFT	100		5,000	956	$29.00	$8.00	$27,724.00
SONY	100		5,000	9,201	$20.00	$4.00	$184,020.00
		0					$213,791.00

Figure CS18.4 The optimization sheet.

The trade-off curve sheet presents the change in the overall revenue as the user changes the upper bound on the total number of calls to be made. (See Figure CS18.5.) The user varies these values by specifying a range and a step size for the trial upper bound values. Once this information has been entered, the user can press the “Create Curve” button to view the trade-off curve for these values. He or she can return to the optimization sheet by pressing the “Return to Optimization” button.



Upper Bound on Total Number of Calls	Total Overall Revenue
1,000	$737,477.00
2,000	$737,477.00
3,000	$737,477.00
4,000	$737,477.00
5,000	$737,477.00
6,000	$11,557,749.00
7,000	$11,557,749.00
8,000	$11,557,749.00
9,000	$11,557,749.00
10,000	$11,557,749.00
11,000	$11,557,749.00
12,000	$11,557,749.00
13,000	$11,557,749.00
14,000	$11,557,749.00
15,000	$11,557,749.00
16,000	$11,557,749.00
17,000	$11,557,749.00
18,000	$13,989,157.00
19,000	$13,989,157.00
20,000	$13,989,157.00

Figure CS18.5 The trade-off curve sheet.

Summary

Welcome sheet	Contains the application description and the "Run Demo" and "Start" buttons.
Historical data sheet	Where the user enters the historical input for each customer.
S-curve parameters sheet	Optimizes the s-curve parameters for each customer.
Optimization sheet	Determines the number of calls to make for each customer in order to maximize the total overall profit.
Trade-off curve sheet	Displays the trade-off curve between the overall profit and the upper bound on the total number of calls.

CS18.3 *User Interface*

For this application's user interface, we use navigational and functional buttons, input boxes and message boxes, and a user form. Pressing the "Start" button on the welcome sheet displays an input box for the number of customers to analyze. (See Figure CS18.6.) The user form then ascertains from the user how the historical data will be provided. (See Figure CS18.7.) The user has two options: import a text file, for which he or she will be prompted to select a file to import; or enter the data manually, for which he or she will proceed directly to the historical data sheet. This form therefore requires two option buttons.

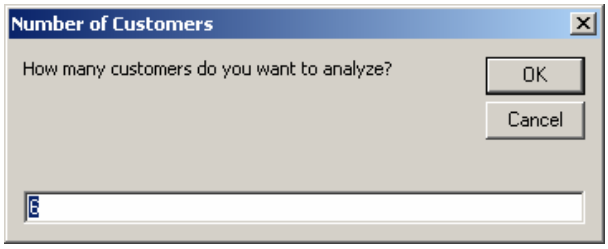


Figure CS18.6 The input box for the number of customers.

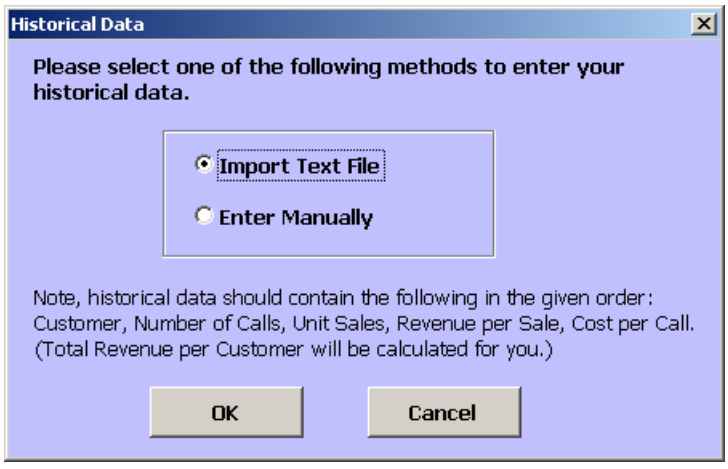


Figure CS18.7 The historical data form.

We include several message boxes on the s-curve parameters sheet to communicate with the user. As each customer's s-curve parameters are optimized, we display a message

box to the user to declare that one customer's parameters are optimized, and so he or she should now enter the input for the next customer. (See Figure CS18.8.)

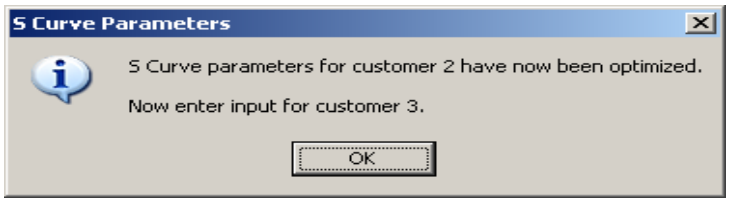


Figure CS18.8 The s-curve input box between customers.

When all of the customers' s-curve parameters have been optimized, a message box appears to inform the user that this sheet is completed. (See Figure CS18.9.)

Figure CS18.9 The s-curve input box when all the customers have been optimized.

We also use message boxes here for two error checks. If the user tries to go to the optimization sheet before optimizing all the customers' s-curve parameters, a message box appears with an error message. (See Figure CS18.10.) If the user continues pressing the "Calculate" button to optimize the customer's s-curve parameters after all the customers' parameters have been optimized, then another error message appears. (See Figure CS18.11.)

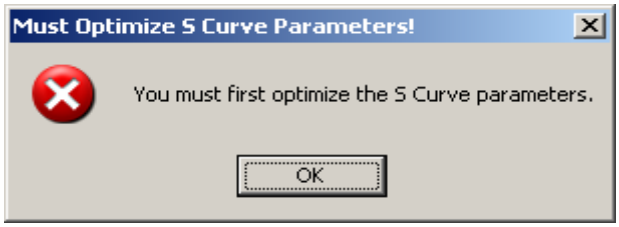


Figure CS18.10 Error checking to finish the s-curve parameters before going to the optimization sheet.

Figure CS18.11 Error checking to prevent further s-curve optimization after is has been completed.

Summary

Input box	Obtains the number of customers to analyze.
Input form	Determines how the user will input the historical data.
Message boxes	Communicates with the user on the s-curve parameter sheet.
Navigational buttons	"End," "Return to Optimization," "View Trade Off Curve."
Functional buttons	"Run Demo," "Start," "Find S-Curve Parameters," "Calculate," "Go To Optimization," "Optimize," "Create Curve."

CS18.4 *Procedures*

We will now outline the procedures for this application beginning with the variable declarations and the initial sub procedures. (See Figure CS18.12.) The "Main" procedure, which is called from the "Start" button, calls the "ClearPrev" procedure to clear the previous values from all the sheets. It then prompts the user with an input box to find the number of customers. Next, it updates all the sheets with this value to prepare the tables. Finally, the procedure ends by displaying the historical data form.

The "RunDemo" procedure is called from the "Run Demo" button on the welcome sheet. (See Figure CS18.13.) It sets a Boolean value and then calls the "Main" procedure. The Boolean value prevents the input box and the user form from being displayed to the user. It then copies the demo data to the historical data sheet and takes the user to this sheet.

The procedures for the historical data form determine if the user will import a text file or enter the historical data directly into the sheet. (See Figure CS18.14.) If the user selects to import a text file, a browser window opens so the user can select his or her file. The file is then imported to the historical data sheet. If the user selects to enter the data manually, then the historical data sheet appears.

The "SCurve" procedure is called when the user presses the "Find S Curve Parameters" button on the historical data sheet. (See Figure CS18.15.) It prepares the s-curve parameters sheet by readying the Solver and copying the input values for the first customer. If the demo is being run, then the demo sales estimates are also copied for the first customer.

The "Calculate" button on the s-curve parameter sheet calls the "CalcS" procedure, which performs error checking and displays message boxes for the user to communicate when customer parameters have been optimized. (See Figure CS18.16.) It runs the Solver, stores the s-curve parameters for each customer, and then copies the input for the next customer.

```
Public NumCust As Integer, UserVal As Variant, i As Integer, _
A() As Double, B() As Double, C() As Double, D() As Double, _
Cust As Integer, iter As Integer, Default As Boolean, _
Max As Double, Min As Double, StepSize As Integer

Sub Main()  'called from Start button
    Application.ScreenUpdating = False
    Call ClearPrev

    'number of customers
    If Default = False Then
        UserVal = InputBox("How many customers do you want to analyze?", "Number of Customers", 6)
        If UserVal = "" Then
            End
        ElseIf IsNumeric(UserVal) = False Then
            MsgBox "Please enter a number."
            Exit Sub
        Else
            NumCust = CInt(UserVal)
        End If
    Else
        NumCust = 6
    End If

    'update sheets
    With Worksheets("HistData")
        For i = 1 To NumCust - 2
            .Range("HistTable").Copy
            .Range("HistTable").Offset(i, 0).Insert Shift:=xlDown
        Next i
        For i = 1 To NumCust
            .Range("HistStart").Offset(i, 0).Value = i
        Next i
        Range(Range("TotalRev").Offset(1, 0), Range("TotalRev").Offset(NumCust, 0)).Name = "HistRevenues"
        Range("HistRevenues").FormulaR1C1 = "=R[]C[-3]*R[]C[-2] - R[]C[-4]*R[]C[-1]"
        Range("HistTotal").Formula = "=SUM(HistRevenues)"
    End With

    With Worksheets("Optimize")
        For i = 1 To NumCust - 2
            .Range("OptTable").Copy
            .Range("OptTable").Offset(i, 0).Insert Shift:=xlDown
        Next i
        For i = 1 To NumCust
            .Range("OptStart").Offset(i, 0).Value = i
        Next i
    End With

    Application.CutCopyMode = False
    Application.ScreenUpdating = True
    If Default = False Then
        frmHistData.Show
    End If
End Sub
```

Figure CS18.12 The variable declarations and the “Main” procedure.

```
Sub RunDemo()   'called from Run Demo button
    Application.ScreenUpdating = False
    Default = True
    Call Main

    'copy demo data
    Worksheets("DemoData").Range("Demo1").Copy
    Worksheets("HistData").Range("HistStart").Offset(1, 0).PasteSpecial

    Worksheets("HistData").Visible = True
    Worksheets("Welcome").Visible = False
    Range("A1").Select

    Application.ScreenUpdating = True
End Sub

Sub ClearPrev()    'clear rows of previous data
    Worksheets("HistData").Range(Range("HistTable").Offset(1, 0), Range("HistTable").End(xlDown)).Delete
    Worksheets("HistData").Range(Range("HistTable"), Range("HistTable").Offset(-1, 0)).ClearContents

    Worksheets("Optimize").Range(Range("OptTable").Offset(1, 0), Range("OptTable").End(xlDown)).Delete
    Worksheets("Optimize").Range(Range("OptTable"), Range("OptTable").Offset(-1, 0)).ClearContents
    Worksheets("Optimize").Range("OptUB").ClearContents

    Worksheets("TradeOff").Range(Range("TradeStart").Offset(1, 0), _
        Range("TradeStart").Offset(1, 1).End(xlDown)).ClearContents
    Worksheets("TradeOff").Range("MinUB").ClearContents
    Worksheets("TradeOff").Range("MaxUB").ClearContents
    Worksheets("TradeOff").Range("StepUB").ClearContents
End Sub
```

Figure CS18.13 The “RunDemo” and “ClearPrev” procedures.

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

Private Sub cmdOK_Click()
    If optImport Then
        'open browse window to import file
        Dim pathname As String
        pathname = Application.GetOpenFilename("TEXT (*.txt), *.txt", , "Historical Data")
        If pathname = "False" Then
            Exit Sub
        End If

        'go to historical data sheet
        Worksheets("HistData").Visible = True
        Worksheets("Welcome").Visible = False
        Worksheets("HistData").Activate

        'import text file
        With ActiveSheet.QueryTables.Add(Connection:= _
            "TEXT;" & pathname & "", Destination:=Range("HistStart").Offset(1, 0))

            .PreserveFormatting = True
            .RefreshStyle = xlOverwriteCells
            .AdjustColumnWidth = False

            .TextFileStartRow = 1
            .TextFileParseType = xlDelimited
            .TextFileCommaDelimiter = True

            .Refresh BackgroundQuery:=False
        End With

    ElseIf optManual Then
        'go to historical data sheet
        Worksheets("HistData").Visible = True
        Worksheets("Welcome").Visible = False

        MsgBox "You may now enter your data into the table."
    End If

    Unload Me
End Sub
```

Figure CS18.14 The historical data form procedures.

The “GoToOpt” procedure is called from the “Go To Optimization” button on the s-curve parameter sheet. (See Figure CS18.17.) It prepares the optimization sheet by copying the customer names, revenues, and costs and naming several ranges for the Solver model. It then inserts formulas for the sales using the s-curve parameters found in the previous procedure. It also inserts a formula to calculate the total overall profit for all the customers. Next, the Solver is prepared. If the user is running the demo, then the demo value for the upper and lower bounds on the number of calls per customer are also inserted in the sheet.

```
Sub SCurve()    'prepare S Curve sheet
    Application.ScreenUpdating = False
    Worksheets("SCurve").Visible = True
    Worksheets("HistData").Visible = False

    'prep Solver
    SolverReset
    SolverOK SetCell:=Range("ObjFunc"), MaxMinVal:=2, ByChange:=Range("DecVar")
    SolverAdd CellRef:=Range("Aparam"), Relation:=1, FormulaText:=Range("BParam")
    SolverOptions AssumeNonNeg:=True

    Cust = 1
    ReDim A(NumCust), B(NumCust), C(NumCust), D(NumCust)

    'copy data for first customer
    Range("Cust").Value = Worksheets("HistData").Range("HistStart").Offset(Cust, 0).Value
    Range("CustRev").Value = Worksheets("HistData").Range("HistRev").Offset(Cust, 0).Value
    Range("CustCost").Value = Worksheets("HistData").Range("HistCost").Offset(Cust, 0).Value

    Range("NumCalls").Value = Worksheets("HistData").Range("HistCalls").Offset(Cust, 0).Value
    Range("Sales").Value = Worksheets("HistData").Range("HistSales").Offset(Cust, 0).Value

    If Default Then 'copy estimates for first customer
        Worksheets("DemoData").Range(Range("DemoS").Offset(Cust, 1), Range("DemoS").Offset(Cust, 5)).Copy
        Range("SalesS").Offset(1, 0).PasteSpecial xlPasteValues, , , True
    End If

    Range("A1").Select
    Application.ScreenUpdating = True
End Sub
```

Figure CS18.15 The “SCurve” procedure.

```
Sub CalcS() 'called from Calculate button on S Curve sheet
    If Cust > NumCust Then
        MsgBox "All customer's parameters have already been optimized.", vbCritical, "S Curve Parameters Done!"
        Exit Sub
    End If

    If Default Then 'copy estimates
        Worksheets("DemoData").Range(Range("DemoS").Offset(Cust, 1), Range("DemoS").Offset(Cust, 5)).Copy
        Range("SalesS").Offset(1, 0).PasteSpecial xlPasteValues, , , True
    End If

    Application.ScreenUpdating = False
    'starting guess: a = 0, c = 0.5
    Range("AParam").Value = 0
    Range("BParam").ClearContents
    Range("CParam").Value = 0.5
    Range("DParam").ClearContents

    'run Solver
    SolverSolve UserFinish:=True
    SolverFinish KeepFinal:=True

    'store parameters
    A(Cust) = Range("DecVar").Cells(1)
    B(Cust) = Range("DecVar").Cells(2)
    C(Cust) = Range("DecVar").Cells(3)
    D(Cust) = Range("DecVar").Cells(4)

    If Cust = NumCust Then
        MsgBox "S Curve parameters for all customers have now been optimized.", vbInformation, "S Curve Parameters Done!"
        Cust = Cust + 1
    Else
        MsgBox "S Curve parameters for customer " & Cust & " have now been optimized." & _
            vbCrLf & vbCrLf & "Now enter input for customer " & Cust + 1 & ".", vbInformation, "S Curve Parameters"
        Cust = Cust + 1

        'copy new data
        Range("Cust").Value = Worksheets("HistData").Range("HistStart").Offset(Cust, 0).Value
        Range("CustRev").Value = Worksheets("HistData").Range("HistRev").Offset(Cust, 0).Value
        Range("CustCost").Value = Worksheets("HistData").Range("HistCost").Offset(Cust, 0).Value

        Range("NumCalls").Value = Worksheets("HistData").Range("HistCalls").Offset(Cust, 0).Value
        Range("Sales").Value = Worksheets("HistData").Range("HistSales").Offset(Cust, 0).Value
    End If
    Application.ScreenUpdating = True
End Sub
```

Figure CS18.16 The “CalcS” procedure.

```
Sub GoToOpt()   'called from Go to Optimization button on S Curve sheet; prepare optimization sheet
    If Cust < NumCust Then
        MsgBox "You must first optimize the S Curve parameters.", vbCritical, "Must Optimize S Curve Parameters!"
        Exit Sub
    End If
    Application.ScreenUpdating = False
    Worksheets("Optimize").Visible = True
    Worksheets("SCurve").Visible = False

    'copy customer names, revenues, and costs
    Range(Range("HistStart").Offset(1, 0), Range("HistStart").Offset(NumCust, 0)).Copy
    Range("OptStart").Offset(1, 0).PasteSpecial xlPasteValues
    Range(Range("HistRev").Offset(1, 0), Range("HistRev").Offset(NumCust, 0)).Copy
    Range("OptRev").Offset(1, 0).PasteSpecial xlPasteValues
    Range(Range("HistCost").Offset(1, 0), Range("HistCost").Offset(NumCust, 0)).Copy
    Range("OptCost").Offset(1, 0).PasteSpecial xlPasteValues

    'name ranges
    Range(Range("OptCalls").Offset(1, 0), Range("OptCalls").Offset(NumCust, 0)).Name = "CallsDV"
    Range(Range("LB").Offset(1, 0), Range("LB").Offset(NumCust, 0)).Name = "LBCon"
    Range("LBCon").Interior.ColorIndex = 2
    Range(Range("UB").Offset(1, 0), Range("UB").Offset(NumCust, 0)).Name = "UBCon"
    Range("UBCon").Interior.ColorIndex = 2
    Range("SumCalls").Formula = "=SUM(CallsDV)"

    'insert formula for sales based on S Curve parameters
    For i = 1 To NumCust
        Range("OptSales").Offset(i, 0).FormulaR1C1 = "=INT(" & A(i) & " + ((" & _
            B(i) & "-" & A(i) & ")*(R[]C[-2])^" & C(i) & ")/(" & D(i) & "+ (R[]C[-2])^" & C(i) & "))"
        Range("OptTotal").Offset(i, 0).FormulaR1C1 = "=R[]C[-3]*R[]C[-2] - R[]C[-5]*R[]C[-1]"
    Next i

    'calc total overall
    Range("OptOF").Formula = "=SUM(" & Range(Range("OptTotal").Offset(1, 0), _
        Range("OptTotal").Offset(NumCust, 0)).Address & ")"

    'prep Solver
    With Worksheets("Optimize")
        SolverReset
        SolverOK SetCell:=Range("OptOF"), MaxMinVal:=1, ByChange:=Range("CallsDV")
        SolverAdd CellRef:=Range("CallsDV"), Relation:=3, FormulaText:=Range("LBCon")
        SolverAdd CellRef:=Range("CallsDV"), Relation:=1, FormulaText:=Range("UBCon")
        SolverAdd CellRef:=Range("SumCalls"), Relation:=1, FormulaText:=Range("OptUB").Address
        SolverAdd CellRef:=Range("CallsDV"), Relation:=4    'integer values
        SolverOptions AssumeNonNeg:=True, Convergence:=False
    End With

    If Default Then 'default bound values
        Range("OptUB").Value = 10000
        Range("LBCon").Value = 100
        Range("UBCon").Value = 5000

        With Worksheets("TradeOff")
            .Range("MinUB").Value = 1000
            .Range("MaxUB").Value = 20000
            .Range("StepUB").Value = 1000
        End With
    End If

    Range("A1").Select
    Application.ScreenUpdating = True
End Sub
```

Figure CS18.17 The “GoToOpt” procedure.

The “Optimize” procedure is called from the “Optimize” button on the optimization sheet. (See Figure CS18.18.) It places an initial guess in the decision variable cells and then runs the Solver.

The “TradeOff” procedure is called from the “Create Curve” button on the trade-off curve sheet. (See Figure CS18.18.) This procedure performs a loop from the minimum trial value to the maximum trial value using the step size specified by the user for the upper bound value on the total number of calls. In this loop, the upper bound value is updated and the “Optimize” procedure is re-called to re-solve the model using the Solver. When the loop is completed, the trade-off curve chart is updated.

```
Sub Optimize()  'called from Optimize button on optimization sheet
    'initial guess
    For Cust = 1 To NumCust
        Range("OptCalls").Offset(Cust, 0).Value = Range("UB").Offset(Cust, 0).Value - 50
    Next

    'run Solver
    SolverSolve USerFinish:=True
End Sub

Sub TradeOff()  'called from Create Curve button on trade-off sheet
    Range(Range("TradeStart").Offset(1, 0), Range("TradeStart").Offset(1, 1).End(xlDown)).ClearContents
    Worksheets("Optimize").Visible = True

    'change UB value and run solver
    iter = 1
    Max = Range("MaxUB").Value
    Min = Range("MinUB").Value
    StepSize = Range("StepUB").Value
    For i = Min To Max Step StepSize
        Range("OptUB").Value = i
        Worksheets("Optimize").Activate
        Call Optimize

       Worksheets("TradeOff").Range("TradeStart").Offset(iter, 0).Value = i
       Worksheets("TradeOff").Range("TradeStart").Offset(iter, 1).Value = Range("OptOF").Value
       iter = iter + 1
    Next i

    Worksheets("Optimize").Visible = False
    Range(Range("TradeStart").Offset(1, 0), Range("TradeStart").Offset(1, 1).End(xlDown)).Name = "TradeGraph"
    ActiveSheet.ChartObjects(1).Select
    ActiveChart.SetSourceData Source:=Range("TradeGraph")
    Range("A1").Select
End Sub
```

Figure CS18.18 The “Optimize” and “TradeOff” procedures.

```
Sub EndProg()
     Worksheets("Welcome").Visible = True
     ActiveSheet.Visible = False
     Default = False
End Sub

Sub ReturnOpt()
     Worksheets("Optimize").Visible = True
     ActiveSheet.Visible = False
End Sub

Sub ViewTrade()
     Worksheets("TradeOff").Visible = True
     ActiveSheet.Visible = False
End Sub
```

Figure CS18.19 The navigational procedures.

Figure CS18.19 displays the navigational procedures.

Summary

Main	Initializes the application and prompts the user with the input box and historical data form.
RunDemo	Copies the demo data to the historical data sheet.
ClearPrev	Clears the previous values from all the sheets.
Historical data form procedures	Either import the text file or take the user to the historical data sheet to enter data manually.
SCurve	Prepares the s-curve parameters sheet values and the Solver model.
CalcS	Optimizes the s-curve parameters for the current customer.
GoToOpt	Prepares the optimization sheet values and the Solver model.
Optimize	Solves the optimization model.
TradeOff	Loops over the user’s upper bound values and records the optimized profit values for the trade-off curve.
Navigational procedures	For the “End,” “Return to Optimization,” and “View Trade-Off Curve” buttons.

CS18.5 *Re-solve Options*

The user can re-solve this application by re-optimizing the model on the optimization sheet or re-creating the trade-off curve on the trade-off curve sheet. To re-solve the optimization model, the user simply changes the upper and lower bounds for the number of calls per customer and presses the "Optimize" button again. To re-create the trade-off curve, the user changes the interval for the trial values of the upper bound on the total number of calls and presses the "Create Curve" button again.

Summary

"Optimize" button	Can be used to re-optimize the optimization model.
"Create Curve" button	Can be used to re-create the trade-off curve.

CS18.6 *Summary*

- This application seeks to determine the optimal allocation of sales force in order to maximize profits.
- This application requires five worksheets: the welcome sheet, the historical data sheet, the s-curve parameters sheet, the optimization sheet, and the trade-off curve sheet.
- For this application's user interface, we use navigational and functional buttons, input boxes and message boxes, and a user form.
- Several application procedures optimize the s-curve parameters, optimize the main model, and create a trade-off curve.
- The user can re-solve this application by re-optimizing the model on the optimization sheet or by re-creating the trade-off curve on the trade-off curve sheet.

CS18.7 *Extensions*

- Re-design the s-curve sheet such that the values for all the customers appear at the same time on the sheet. In other words, locate the estimates, input, calculations, errors, and parameters in one table for each customer on the sheet. Allow the user to enter the sales estimates and optimize over all the customers as often as desired; in this way, the user may refine his or her guesses for some or all of the customers.
- Add a cost curve and a revenue curve to the trade-off curve. In other words, for each upper bound trial value, not only record the total profit, but record the total costs and revenues.
- Add error checking to ensure reasonable bounds are given. (Note that without bounds the program will run forever; ensure that this cannot happen.)