2009

EUM Final Project

Weather Derivatives – Real Time Simulation

Greg Critchley, Alison Howlett, Marc Lantos, Graham Page
Niklas Selander, Mike Tang, and Chen Xu

4/9/2009

What is it?

Our program is one that allows users to enter the details of their weather derivative contract, and will – in real time – calculate the expected payout of that contract using historical data. Users can enter either a futures contract, or an options contract. Other inputs include: the option price; the month, city and province in which the derivative contract will be used; the number of years of historical data to use; whether the contract is a Heating Degree Day contract, or a Cooling Degree Day contract; the threshold temperature after which each degree will constitute a HDD or CDD; the revenue/cost per HDD/CDD (e.g. $10 / HDD); and finally, the strike number of HDDs or CDDs (i.e. the number of HDDs/CDDs before any payouts will occur).

Using the month, city, province, and number of historical years, the program will connect to Environment Canada, download the appropriate historical data (where available), and use it to calculate the average expected payout, as well as the minimum, maximum and standard deviation. The simulation is performed using the Bootstrap method which is explained below.

After all calculations are performed, a page containing the inputs, calculations and three graphs is generated. The first graph is a chart of the daily average temperatures for each of the most recent five years, as well as a 5 year average. The second graph is a chart of the average temperature for the month, for each of the years selected. The final chart is a display of the simulation distribution, and includes the payout and cumulative probabilities.

This page also contains four buttons; delete sheet, view raw data, print results, and compare simulations. The first button deletes the sheet when it is no longer necessary. The second button will generate a page containing all of the raw data used in the simulation. This includes the raw weather data, the payouts for each of the months, and all of the simulated data. Because of the length of time it takes to print data only 100 simulations are printed but, if the user wishes, they can click on "Display All Trial Data" which will load all 10,000 simulations. The print results button sets the appropriate print area and launches the print dialog box so that users may print the results of the simulation. Finally, the compare simulations button is useful when the user has run several simulations. It will take the key statistics (i.e. average, min, max and standard deviation) from each of the simulations that have been run (and not deleted) and will display them on a new page where they can be compared.

Comparing Weather Derivative Valuation Models

1. Bootstrap Method

The bootstrap method selects daily temperature values from the population of historical weather data selected, and then it runs through the monthly payout. The program is currently set to pull the inputted number of years of data (therefore, population size equals number of years), from which 10,000 daily temperature values will be pulled randomly and run through the monthly payout function. The final payout figure outputted is the average of the 10,000 payouts.

2. Monte Carlo Method

The Monte Carlo method is useful for modeling phenomena with unknown significant uncertainty in inputs. It would be used to simulate the payout of the weather derivatives from a normal distribution. If Monte Carlo was used, it would require a regression of different variables against the temperature to determine a function that would predict the temperature. This regression equation would then be run 10,000 times to achieve different temperatures that could fit to a normal distribution. These 10,000 estimated temperatures would then be passed through the monthly payout function. The limitation of this method for weather is that we do not know what significant variables would produce a high correlation (close to 1) with the change in weather temperature to create a regression function. Historical weather data is most likely the variable that will be significant and produce the highest correlation; in that case the bootstrap method is the most efficient.

Adjusting Inputs

Under the Weather UserForm code, the maximum, minimum and change amount can be adjusted in the code. The input boxes are titled the following:
Text Box and Spin Box 1 – Years
Text Box and Spin Box 2 – Threshold HDD/CDD
Text Box and Spin Box 3 – Cost/Revenue per HDD/CDD
Text Box and Spin Box 4 – Strike
Text Box and Spin Box 5 – Instrument Price

```
   'TEXT BOX & SPIN BOX 4 (strike)

   TextBox4.Locked = True
   SpinButton4.MAX = 100000 – adjust this value to change the maximum
   SpinButton4.MIN = 0 – adjust this value to change the minimum
   SpinButton4.Value = 50 – adjust this value to change the default value
   SpinButton4.SmallChange = 25 – adjust this to change the amount by which the value
above increases or decreases
   TextBox4.Value = SpinButton4.Value
```

Summary of Functions/Subs

Sub ADD_BUTTON: The purpose of this macro is to create the buttons (it is called in other macros). The size, coordinates, name, sheet and macro to call are passed to the sub, and it then creates a corresponding button.

Function GET_AXIS_SIZE: The purpose of this function is to calculate the correct scale of the y axis for each chart, so that it perfectly fills the chart. It is passed the data array as well as any limitations to be used (i.e. row or column limit). It is then used when generating each of the charts.

Sub DAILY_CHART: The purpose of this macro is to generate a chart that shows the daily temperatures of the selected city for the previous 5 years and includes the 5 year average.

Sub AVG_CHART: The purpose of this macro is to generate a chart that shows the average temperatures of the selected city for the selected month over the entire time period.

Sub SIM_CHART: The purpose of this macro is to generate a chart that shows the simulation distribution (payout probability and cumulative probability).

Sub COMPARE_RESULTS: The purpose of this macro is to compare the results of the two or more derivative simulation calculations. When called, it collects the mean, min, max and standard deviation results from each of the simulations conducted, and displays them on a new sheet for user comparison.

Sub Submit: This macro is called after users enter all of the required information in to the user form and click "calculate". It calls various functions and subs to collect the weather data, calculate the payout, generate the charts, format the sheets, and so forth.

Sub FormatSheetMain: This macro formats the sheet that contains the original output.

Sub FormatRawData: This macro formats the sheet that contains the raw data.

Sub FormatComparison: This macro formats the sheet that contains the comparisons of the outputs.

Function HEDGE_PAYOUT: This function does all calculations and then stores the statistics, inputs and outputs in a nested array. These calculations include the future/option payout; the average, standard deviation, minimum and maximum.

```
Function HEDGE_PAYOUT(ByVal PROV As Variant, _
   ByVal CITY As Variant, _
   ByVal MONTH As Variant, _
   ByVal NUMYEARS As Integer, _
   ByVal HDDCDD As Variant, _
   ByVal CTYPE As Variant, _
   ByVal THRESH_TEMP As Variant, _
   ByVal DOLLAR_PER As Double, _
   ByVal STRIKE As Variant, _
   Optional ByVal OPTION_PRICE As Variant)
On Error GoTo ERROR_LABEL

'Variable List
'HDDCDD - Contract is HDD or CDD
'CTYPE - Future or Option
```

```
'THRESH_TEMP - HDD/CCD is one degree Above/Below this temp
'DOLLAR_PER - Number of dollars per degree over STRIKE
'STRIKE - Number of degrees under/over THRESH_TEMP before payouts begin
'OPTION_PRICE - Cost of option (if applicable)

Dim h As Variant
Dim i As Variant
Dim j As Variant
Dim k As Variant

Dim WEATHER_DATA As Variant
Dim NUMROWS As Integer
Dim HDDCOUNT As Long
Dim CDDCOUNT As Long
Dim TOTALHDD As Long
Dim TOTALCDD As Long
Dim TEMP_DATA As Variant
Dim PAYOUT_ARRAY As Variant
Dim TOTAL_P As Variant
Dim AVG_P As Variant
Dim TOTAL_ARRAY As Variant
Dim STDEV As Variant
Dim MIN As Variant
Dim MAX As Variant
Dim OUTPUT_ARRAY As Variant
Dim STATS_ARRAY As Variant
```

1. See if Option Price was entered

```
If CTYPE = "Option" Then
   If IsMissing(OPTION_PRICE) Then
      HEDGE_PAYOUT = "Need Option Price"
      Exit Function
   End If
   If OPTION_PRICE = "" Then
      HEDGE_PAYOUT = "Need Option Price"
      Exit Function
   End If
End If
```

2. Get Weather Data

```
WEATHER_DATA = GET_WEATHER(PROV, CITY, MONTH, NUMYEARS)
If IsArray(WEATHER_DATA) = False Then
   ReDim OUTPUT_ARRAY(1 To 4)
   OUTPUT_ARRAY(1) = "No Data"
   OUTPUT_ARRAY(2) = "No Data"
   OUTPUT_ARRAY(3) = "No Data"
   OUTPUT_ARRAY(4) = "No Data"
   HEDGE_PAYOUT = OUTPUT_ARRAY
   Exit Function
End If
NUMROWS = UBound(WEATHER_DATA, 1)

ReDim PAYOUT_ARRAY(1 To NUMROWS)
'Determine Contract Payout per historical payout
For h = 1 To NUMROWS
   TOTALHDD = 0
   TOTALCDD = 0
```

3. Determine Degree Days in Month

```
   For i = 1 To 31
      TEMP_DATA = WEATHER_DATA(h, i)
      If TEMP_DATA = "" Then GoTo SkipIt

      HDDCOUNT = THRESH_TEMP - TEMP_DATA
      If HDDCOUNT < 0 Then HDDCOUNT = 0

      CDDCOUNT = TEMP_DATA - THRESH_TEMP
      If CDDCOUNT < 0 Then CDDCOUNT = 0

      TOTALHDD = TOTALHDD + HDDCOUNT
      TOTALCDD = TOTALCDD + CDDCOUNT
SkipIt:
   Next i
```

4. Determine Payout per historical month

```
   If CTYPE = "Future" Then
      If HDDCDD = "HDD" Then
         PAYOUT_ARRAY(h) = (TOTALHDD - STRIKE) * DOLLAR_PER
      End If
      If HDDCDD = "CDD" Then
         PAYOUT_ARRAY(h) = (TOTALCDD - STRIKE) * DOLLAR_PER
      End If
   ElseIf CTYPE = "Option" Then
      If HDDCDD = "HDD" Then
         PAYOUT_ARRAY(h) = GREG_MAX((TOTALHDD - STRIKE), 0) *
DOLLAR_PER - OPTION_PRICE
      End If
      If HDDCDD = "CDD" Then
```

```
            PAYOUT_ARRAY(h) = GREG_MAX((TOTALCDD - STRIKE), 0) * DOLLAR_PER - OPTION_PRICE
        End If
    End If
Next h
```

5. Simulate future/option payout

```
k = 10000
TOTAL_ARRAY = SIMULATION(k, PAYOUT_ARRAY)
```

6. Calculate Average

```
TOTAL_P = 0
For j = 1 To k
    TOTAL_P = TOTAL_P + TOTAL_ARRAY(j)
Next j
AVG_P = TOTAL_P / k
```

7. Calculate STDEV, Min, and Max

```
MIN = TOTAL_ARRAY(1)
MAX = TOTAL_ARRAY(1)
For j = 1 To k
    STDEV = STDEV + (TOTAL_ARRAY(j) - AVG_P) ^ 2
    If TOTAL_ARRAY(j) < MIN Then MIN = TOTAL_ARRAY(j)
    If TOTAL_ARRAY(j) > MAX Then MAX = TOTAL_ARRAY(j)
Next j
STDEV = Sqr(STDEV / (k - 1))
```

8. Store stats in an array

```
ReDim STATS_ARRAY(1 To 4)
STATS_ARRAY(1) = AVG_P
STATS_ARRAY(2) = MIN
STATS_ARRAY(3) = MAX
STATS_ARRAY(4) = STDEV
```

9. Store outputs in an array

```
ReDim OUTPUT_ARRAY(1 To 4)
OUTPUT_ARRAY(1) = STATS_ARRAY
OUTPUT_ARRAY(2) = PAYOUT_ARRAY
OUTPUT_ARRAY(3) = TOTAL_ARRAY
OUTPUT_ARRAY(4) = WEATHER_DATA

HEDGE_PAYOUT = OUTPUT_ARRAY
```

```
Exit Function
ERROR_LABEL:
HEDGE_PAYOUT = Err.Number
End Function
```

Function HISTOGRAM: This function is used for generating the data array that is used when creating the histogram chart.

Function CHECK_VALUE: This simple functions check to see whether or not a string is blank.

Sub form: The purpose of this macro is to make the weather derivative input form load.

Sub REMOVE_SHEET: The function of this macro is to remove data sheets (i.e. the selected sheet) at the user's request. It is used in other macros and with the ADD_BUTTON macro.

Sub PRINT_RESULTS: The purpose of this macro is to format the instrument details, the payout data and the graphs that are generated as a result of the of the weather data simulation, in a printer-friendly manner and then print. This macro is used in other macros that create the "Print Results" button.

Function CALL_PUBDATA: This function is used to store data in a Public array using a method that works with both Excel 2003 and Excel 2007. It is used for storing the raw data, so that it call be displayed on user request. The sheet name of the simulation is a mandatory input, with all of the pieces of raw data being optional. If only the sheet name is entered, the function will return the raw data corresponding to that sheet name. However, if a sheet name and raw data are passed (done during the simulation), then the function will store that data in a public array. Technically the array is private to that module, but its effect is that the data is persistent.

Sub GET_RAW: This macro retrieves the raw data if the user wishes to view it. It also includes a progress bar that shows the user how much longer it will take to retrieve the data. It calls the macro FormatRawData that organizes the data in a user-friendly manner. It also calls the macro ADD_BUTTON that creates the buttons "Display All Trial Data" and "Delete Sheet". It displays the payout data, trial data, and weather data. This macro can be called in two ways. It can either be called with a portion of all of the simulations displayed, or with all simulations displayed. See below.

Sub GET_RAW_SMALL: The purpose of this function is to call GET_RAW using a smaller sample of raw data (100 trials). This is the default.

Sub GET_RAW_FULL: This macro creates a message box that asks the user if they would like to see all the raw data and trial simulations. Generally, only 100 simulations are visible to the user when they 'ask' to see the raw data, but this module allows them to see all 10,000 simulation trials. It is used in conjunction with Sub GET_RAW to retrieve the raw data.

Function SIMULATION: The purpose of this function is to run a simulation of the probability of receiving a payout on the weather derivative. The input is the number of trials for the simulation (NUMTRIALS), which is set to 10,000, and the monthly payout data array (DATA_ARRAY). One of the monthly payouts from historical data are randomly selected 10,000 times, and each trial is stored in an output array.

Function GET_WEATHER: The purpose of this function is to retrieve the data from the internet. On the user form, the user chooses the province (PROV) and is given a corresponding list of cities from which to choose (CITY). All of these cities have individual number codes which are used to find the appropriate webpage on Environment Canada. The other user-selected input is the month (MONTH), which also have corresponding numerical values, that the user will use to base their weather derivative on. The final input is the number of years (NUMYEARS) for which the user wants to retrieve the data. The data that is retrieved is the average daily temperature, which is put in an array and returned to the function/sub that called it.

1. Assign correct city variables

```
'ALBERTA
If PROV = "ALBERTA" Then
   If CITY = "BANFF" Then CITY = 27378
   If CITY = "CALGARY" Then CITY = 2205
   If CITY = "CORONATION" Then CITY = 27212
   If CITY = "EDMONTON" Then CITY = 1865
   If CITY = "EDSON" Then CITY = 31588
   If CITY = "FORT MCMURRAY" Then CITY = 31288
   If CITY = "GRANDE PRAIRIE" Then CITY = 27188
   If CITY = "HIGH LEVEL" Then CITY = 2726
   If CITY = "JASPER" Then CITY = 10223
   If CITY = "LETHBRIDGE" Then CITY = 31447
   If CITY = "LLOYDMINSTER" Then CITY = 1920
   If CITY = "MEDICINE HAT" Then CITY = 44647
   If CITY = "PEACE RIVER" Then CITY = 2770
   If CITY = "PINCHER CREEK" Then CITY = 8791
   If CITY = "RED DEER" Then CITY = 2134
   If CITY = "SLAVE LAKE" Then CITY = 31528
End If
```

……………………..

The actual code contains many more cities, but they have been removed for illustrative purposes

…………………….

2. Assign correct province variables

```
If PROV = "ALBERTA" Then PROV = "AB"
If PROV = "BRITISH COLUMBIA" Then PROV = "BC"
If PROV = "MANITOBA" Then PROV = "MB"
If PROV = "NEW BRUNSWICK" Then PROV = "NB"
If PROV = "NEWFOUNDLAND" Then PROV = "NF"
If PROV = "NORTHWEST TERRITORIES" Then PROV = "NT"
If PROV = "NOVA SCOTIA" Then PROV = "NS"
If PROV = "NUNAVUT" Then PROV = "NU"
If PROV = "ONTARIO" Then PROV = "ON"
If PROV = "PRICE EDWARD ISLAND" Then PROV = "PE"
If PROV = "QUEBEC" Then PROV = "QC"
If PROV = "SASKATCHWAN" Then PROV = "SK"
If PROV = "YUKON TERRITORY" Then PROV = "YT"
```

3. Assign correct month variables

```
If MONTH = "January" Then MONTH = 1
If MONTH = "February" Then MONTH = 2
If MONTH = "March" Then MONTH = 3
If MONTH = "April" Then MONTH = 4
If MONTH = "May" Then MONTH = 5
If MONTH = "June" Then MONTH = 6
If MONTH = "July" Then MONTH = 7
If MONTH = "August" Then MONTH = 8
If MONTH = "September" Then MONTH = 9
If MONTH = "October" Then MONTH = 10
If MONTH = "November" Then MONTH = 11
If MONTH = "December" Then MONTH = 12
```

4. Download all the weather data

```
CURRENT_YEAR = Year(Now()) - 1
ReDim WEATHER_DATA(1 To NUMYEARS)

For h = 1 To NUMYEARS
   SOURCE_URL =
"http://www.climate.weatheroffice.ec.gc.ca/climateData/dailydata_e.html?timeframe=2&
Prov=" & PROV & "&StationID=" & CITY & "&Year=" & CURRENT_YEAR &
"&Month=" & MONTH
   WEATHER_DATA(h) = SAVE_WEB_DATA_PAGE_FUNC(SOURCE_URL)
   WEATHER_DATA(h) = Replace(WEATHER_DATA(h), Chr(10), "")
   CURRENT_YEAR = CURRENT_YEAR - 1
   PROG.ProgressBar1.Value = GREG_MIN(PROG.ProgressBar1.Value + (1 /
NUMYEARS) * 100, 100)
   DoEvents
```

```
Next h

Unload PROG
```

5. Trim all of the useless HTML code and remove irretrievable years

```
ReDim TRIMMED_DATA(1 To 1)
k = 1
For i = 1 To NUMYEARS
   TEMP_ARRAY = WEATHER_DATA(i)
   CHECK_STR = "<tr id=" & Chr(34) & "dataTableOddRow" & Chr(34) & "><td id="
& Chr(34) & "dataTableRowHeader" & Chr(34) & ">"

   j = InStr(1, WEATHER_DATA(i), CHECK_STR)
   If j = 0 Then GoTo SkipIt

   CHECK_STR = "<td id=" & Chr(34) & "dataTableRowData" & Chr(34) &
"></td><td id=" & Chr(34) & "dataTableRowData" & Chr(34) & "></td><td id=" &
Chr(34) & "dataTableRowData" & Chr(34) & "></td>"
   If InStr(j, WEATHER_DATA(i), CHECK_STR) <> 0 Then GoTo SkipIt

   CHECK_STR = "<abbr title=" & Chr(34) & "Summary" & Chr(34) & " lang=" &
Chr(34) & "en" & Chr(34) & ">Sum</abbr></td>"
   m = InStr(j, WEATHER_DATA(i), CHECK_STR)
   If m = 0 Then GoTo SkipIt
   n = Len(CHECK_STR)

   TRIMMED_DATA(k) = Mid(WEATHER_DATA(i), j, m - j + n)

   k = k + 1
   If k <= NUMYEARS Then ReDim Preserve TRIMMED_DATA(1 To k)
SkipIt:
Next i
If k = 1 Then
   GET_WEATHER = "No Data"
   Exit Function
End If
k = k - 1
'k is now the number of years used
```

6. Get the data out of each page (i.e. daily temperatures for each month)

```
ReDim DATA_ARRAY(1 To k, 1 To 31)
For o = 1 To k
   q = 1
   For s = 1 To 31
```

```
    'Get Average Daily Temperature
    CHECK_STR = "<td id=" & Chr(34) & "dataTableRowData" & Chr(34) & ">"
    n = Len(CHECK_STR)
    p = InStr(q, TRIMMED_DATA(o), CHECK_STR)
    p = InStr(p + n, TRIMMED_DATA(o), CHECK_STR)
    p = InStr(p + n, TRIMMED_DATA(o), CHECK_STR)
    If p = 0 Then Exit For

    r = InStr(p + n, TRIMMED_DATA(o), "<")
    If r = 0 Then Exit For
    DATA_ARRAY(o, s) =
CONVERT_STRING_NUMBER_FUNC(Trim(Mid(TRIMMED_DATA(o), p + n, r - (p
+ n))))
    If DATA_ARRAY(o, s) = "&nbsp;" Then DATA_ARRAY(o, s) = ""
    If IsNumeric(DATA_ARRAY(o, s)) = False Then DATA_ARRAY(o, s) = ""

    'Advance q to next row of data
    CHECK_STR = "</td></tr><tr id="
    n = Len(CHECK_STR)
    q = InStr(r, TRIMMED_DATA(o), CHECK_STR)
    If q = 0 Then Exit For
```

7. Exit Loop when there is no more data

```
    CHECK_STR = Chr(34) & "dataTableEvenRow" & Chr(34) & "><td id=" &
Chr(34) & "dataTableRowHeader" & Chr(34) & "><abbr title=" & Chr(34) &
"Summary"
    t = Len(CHECK_STR)
    If Mid(TRIMMED_DATA(o), q + n, t) = CHECK_STR Then
      For u = s + 1 To 31
        DATA_ARRAY(o, u) = ""
      Next u
      Exit For
    End If

    CHECK_STR = Chr(34) & "dataTableOddRow" & Chr(34) & "><td id=" &
Chr(34) & "dataTableRowHeader" & Chr(34) & "><abbr title=" & Chr(34) &
"Summary"
    t = Len(CHECK_STR)
    If Mid(TRIMMED_DATA(o), q + n, t) = CHECK_STR Then
      For u = s + 1 To 31
        DATA_ARRAY(o, u) = ""
      Next u
      Exit For
    End If
  Next s
Next o

GET_WEATHER = DATA_ARRAY

Exit Function
ERROR_LABEL:
GET_WEATHER = Err.Number
End Function
```

Other Functions

There were many other functions used to aid in this program's creation, but they were simple and do not need further explanation. Those functions include functions which: calculate min, max, sum and average, calculate the average using row count, calculate the absolute value, convert/transpose arrays, get data from specific row or columns, and remove rows or columns from arrays.