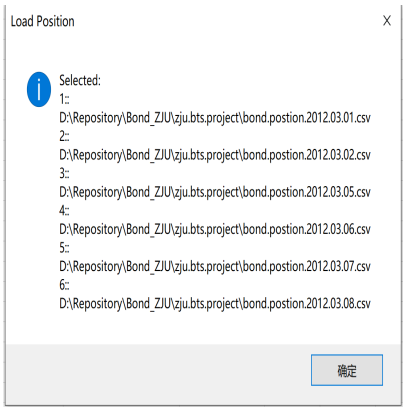
Bond Project

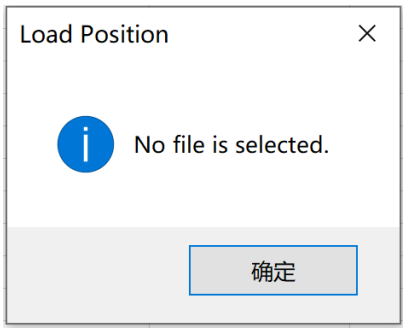
Name: 陈锰
ID: 3170105197
Major: Software Engineering

User Manual

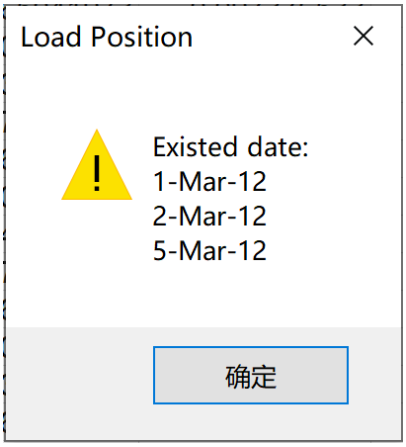
- Load Yield File
  - Open the sheet “Yield.Curve”
  - Click the button “Load_Yield” and select excel files(.csv, .xls, etc) from the file dialog
    - There will be a information-type message box prompting you the selected files



    - There will be a information-type message box prompting you that no file is selected



    - There will be a warning-type message box showing the redundant dates

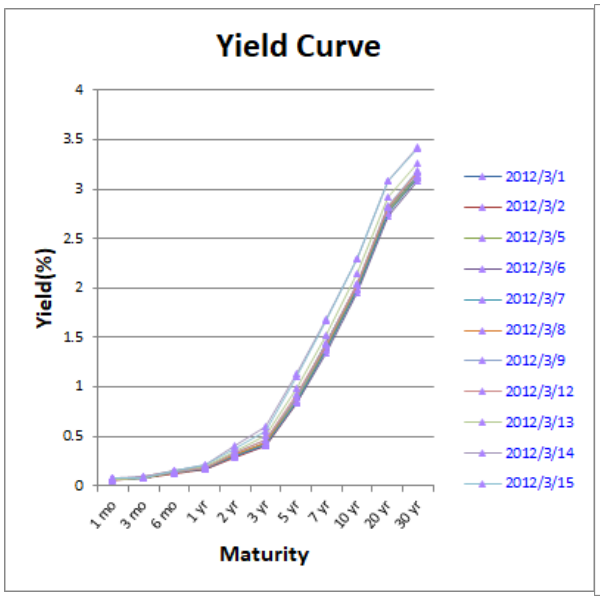


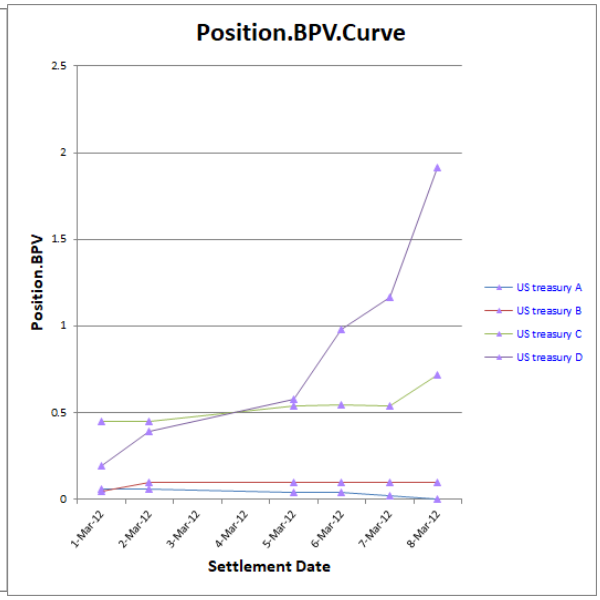
  - Click the button “Plot_Yield” to learning more about yield curves Optional
- Load Position
  - Open the sheet “Bond.Position”

- Click the button “Load_Position” and select excel files(.csv, .xls, etc) from the file dialog
  - There will be a information-type message box prompting you the selected files or no file is selected
  - There will be a warning-type message box showing the redundant dates
- Calculate and Plot Position-BPV
  - Click the button “Calculate_Plot” to obtain the Dirty Price, Accrued Interest, Clean Price, Modified Duration and Position-BPV, then you can find the Position-BPV line chart in the sheet “Position.BPV.Curve”

Note that once you click the button to plot this curve, the former chart will be deleted, thus save the chart in advance if it’s important.

Example Result





Appendix(Code)

Mudule 1

```
Option Explicit
'@Brief: determine whether an item is in the range or not
Function isIn(item As Variant, container As Range) As Boolean
    isIn = Not container.Find(item, LookIn:=xlValues) Is Nothing
End Function
'@Brief: load one csv file from the filefolder and write data to out sheet
'@Return: the redundant date
Function loadFile(path As String, outSheet As Worksheet) As String
    Dim inWb As Workbook
    Set inWb = GetObject(path)
    Dim items As Variant, row As Integer, col As Integer, start As Integer
    start = outSheet.UsedRange.Rows.Count
    For row = 1 To inWb.Sheets(1).UsedRange.Rows.Count
        items = Split(inWb.Sheets(1).Cells(row, 1).Value, "|")
        If isIn(items(0), outSheet.Range("a1:a" & start)) Then
            loadFile = items(0)
            Exit Function
        Else
            For col = 0 To UBound(items)
                'Add Str(20) to the year values of date items
                If InStr(items(col), "-") > 0 Then
```

```
                    items(col) = Mid(items(col), 1, InStrRev(items(col), "-")) &
"20" & Mid(items(col), InStrRev(items(col), "-") + 1)
                End If
                outSheet.Cells(row + start, col + 1).Value = items(col)
            Next
        End If
    Next
    loadFile = ""
End Function
'@Brief: load and deal with files from selected items in the folefolder
Sub loadFiles(sheet As Worksheet, name As String)
    Dim index As Integer, files As String, existed As String
    With Application.FileDialog(msoFileDialogFilePicker)
        .AllowMultiSelect = True
        .Filters.Clear
        .Filters.Add "Excel Files", "*.xls;*.xlw;*.csv"
        .Show

        For index = 1 To .SelectedItems.Count
            files = files & Chr(10) & index & ":: " & .SelectedItems(index)
            existed = existed & Chr(10) & loadFile(.SelectedItems(index), sheet)
        Next
        'Some prompts
        If .SelectedItems.Count > 0 Then
            MsgBox "Selected: " & files, vbOKOnly + vbInformation, name
        Else
            MsgBox "No file is selected.", vbOKOnly + vbInformation, name
        End If
        If Len(Replace(existed, Chr(10), "")) > 0 Then
            MsgBox "Existed date: " & existed, vbOKOnly + vbExclamation, name
        End If
    End With
End Sub
'@Brief: linear interpolation function for fetching yield
Function linearInterpolate(value1 As Double, value2 As Double, rate As Double)
As Double
    linearInterpolate = value1 * (1 - rate) + value2 * rate
End Function
'@Brief: fetch yild form sheet "Yield.Curve"
'@Return: interpolated yield value
Function fetchYield(settlementDate As Date, couponDate As Date) As Double
    Dim row As Range
    'Find the corresponded row with the same date
    Set row = Worksheets("Yield.Curve").Range("a:a").Find(settlementDate,
LookIn:=xlFormulas, lookat:=xlWhole)
    If row Is Nothing Then
        MsgBox "[Error]No such a date in the Sheet Yield.Curve " _
        & Chr(10) & "Date: " & settlementDate _
        , vbOKOnly + vbCritical, "Fetch Yield"
        Exit Function
    End If
    Dim duration
    Dim diff As Double
    Dim index As Integer
    'Maturity duration by month
    duration = Array(1, 3, 6, 12, 24, 36, 60, 84, 120, 240, 360)
    'Calculate the total maturity duration by day
    diff = DateDiff("m", settlementDate, couponDate)
```

```
77      'Find the right position of maturity duration
78      For index = LBound(duration) To UBound(duration)
79          If duration(index) >= diff Then
80              Exit For
81          End If
82      Next
83      Dim startDate As Date, endDate As Date
84      'Find the last date and the next date between which the maturity locates
85      startDate = DateAdd("m", duration(index - 1), settlementDate)
86      endDate = DateAdd("m", duration(index), settlementDate)
87      'Calculate the duration portion
88      diff = DateDiff("d", startDate, couponDate) / DateDiff("d", startDate,
    endDate)
89      'Calculate yield through linear interpolation
90      fetchYield = linearInterpolate(row.Offset(0, index), row.Offset(0, index +
    1), diff) / 100
91      'MsgBox "From " & settlementDate & " To " & couponDate & " Yield: " &
    fetchYield _
92      '& Chr(10) & "C1: " & row.Offset(0, index) & " C2: " & row.Offset(0, index +
    1)
93  End Function
94  '@Brief: calculate duration portion for later use
95  Function calPeriodRatio(settlementDate As Date, couponDate As Date, frequency As
    Integer, mode As Boolean) As Double
96      Dim lastCouponDate As Date, nextCouponDate As Date
97      lastCouponDate = CDate(Year(settlementDate) & "-" & Month(couponDate) & "-"
    & Day(couponDate))
98      While (DateDiff("d", settlementDate, lastCouponDate) > 0)
99          lastCouponDate = DateAdd("m", -12 / frequency, lastCouponDate)
100     Wend
101     nextCouponDate = DateAdd("m", 12 / frequency, lastCouponDate)
102     If mode Then
103         calPeriodRatio = DateDiff("d", settlementDate, nextCouponDate) /
    DateDiff("d", lastCouponDate, nextCouponDate)
104     Else
105         calPeriodRatio = DateDiff("d", lastCouponDate, settlementDate) /
    DateDiff("d", lastCouponDate, nextCouponDate)
106     End If
107 End Function
108 '@Brief: calculate accrued interest
109 Function calAccruedInterest(coupon As Double, frequency As Integer,
    settlementDate As Date, couponDate As Date) As Double
110     calAccruedInterest = coupon / frequency * calPeriodRatio(settlementDate,
    couponDate, frequency, False)
111 End Function
112 '@Brief: calculate a
113 Function calA(frequency As Integer, settlementDate As Date, couponDate As Date)
    As Double
114     calA = calPeriodRatio(settlementDate, couponDate, frequency, True)
115 End Function
116 '@Brief: calculate dirty price and modified duration
117 '@Return: array of PV and MD
118 Function calDirtyPriceAndModifiedDuration(coupon As Double, frequency As
    Integer, settlementDate As Date, couponDate As Date) As Variant
119     Dim yield As Double, cashflow As Double, a  As Double, PV As Double, MD As
    Double
120     Dim i As Integer, m As Integer, D1 As Double, item As Double
121     'Calculate a
```

```
    a = calA(frequency, settlementDate, couponDate)
    'Fetch yield
    yield = fetchYield(settlementDate, couponDate)
    'Calculate number of complete coupon period
    m = Int(DateDiff("m", settlementDate, couponDate) / 12 * frequency)
    'Cashflow in the beginning
    cashflow = coupon / frequency
    'Factor for later use
    D1 = 1 / (1 + yield / frequency)
    PV = 0
    MD = 0
    For i = 0 To m
        item = D1 ^ (a + i)
        PV = PV + item
        MD = MD + (a + i) * item
    Next
    PV = coupon / frequency * PV + 100 * item
    MD = (coupon / frequency * MD + 100 * (a + m) * item) / PV / frequency * D1
    calDirtyPriceAndModifiedDuration = Array(PV, MD)
End Function
```

Sheet “Yield.Curve”

```
Option Explicit

Private Sub Load_Yield_Click()
'Pls ensure the following functions
'1. pop up a window for user to select the file to load
'2. if the data of a particular date exists, a warning msg will be shown
'3. col A to col L will be populated with the data from yc.2012.MM.DD.csv
    loadFiles Worksheets("Yield.Curve"), "Load Yield"
End Sub
'@Brief: plot Yield.Curve
Private Sub Plot_Yield_Click()
    Dim chart As ChartObject, sheet As Worksheet
    Dim row As Integer, endRow As Integer
    Set sheet = Worksheets("Yield.Curve")
    endRow = sheet.UsedRange.Rows.Count
    Application.ScreenUpdating = False
    'Clear the old charts
    If sheet.ChartObjects.Count > 0 Then
        sheet.ChartObjects.Delete
    End If
    'Add a chart at [m5]
    Set chart = sheet.ChartObjects.Add(sheet.[m5].Left, sheet.[m5].Top, 300, 300)
    With chart.chart
        .HasTitle = True
        .ChartTitle.Text = "Yield Curve"
        .ChartTitle.Font.Size = 18
        .HasLegend = True
        .Legend.Font.Size = 8
        .Legend.Font.ColorIndex = 5
        .Legend.position = xlLegendPositionRight
        'Config the Y label
        With .Axes(xlValue, xlPrimary)
            .CrossesAt = .MinimumScale
            .TickLabels.Font.Size = 8
            .HasTitle = True
```

```
            .AxisTitle.Text = "Yield(%)"
            .AxisTitle.Characters.Font.Size = 12
            .AxisTitle.Orientation = xlUpward
        End With
        'Config the X label
        With .Axes(xlCategory)
            .TickLabels.Font.Size = 8
            .TickLabels.Orientation = 50
            .HasTitle = True
            .AxisTitle.Text = "Maturity"
            .AxisTitle.Characters.Font.Size = 12
        End With
        For row = 2 To endRow
            With .SeriesCollection.NewSeries
                .Values = sheet.Range("b" & row & ":l" & row)
                .XValues = sheet.Range("b1:l1")
                .ChartType = xlLineMarkers
                .name = sheet.Range("A" & row).Value
                .MarkerSize = 3
                .Format.Line.Weight = 0.8
                .MarkerStyle = xlMarkerStyleTriangle
                .MarkerForegroundColor = RGB(171, 130, 255)
                .MarkerBackgroundColor = RGB(171, 130, 255)
                .HasDataLabels = False
            End With
        Next
    End With
End Sub
```

Sheet “Bond.Position”

```
Option Explicit

Private Sub Load_Position_Click()
'Pls ensure the following functions
'1. pop up a window for user to select the file to load
'2. if the data of a particular date exists, a warning msg will be shown
'3. col A to col E will be populated with the data from
bond.postion..2012.MM.DD.csv
    loadFiles Worksheets("Bond.Position"), "Load Position"
End Sub

Private Sub Calculate_Plot_Click()
'Pls ensure the following functions
'1. pop up a window for user to select the file to load
'2. if the data of a particular date exists, a warning msg will be shown
'3. col F to col J will be populated
'4. Plot the whole book's Position BPV curve. The book includes 4 bond (A, B, C
and D)
    Dim sheet As Worksheet
    Dim coupon As Double, position As Double
    Dim settlementDate As Date, couponDate As Date
    Dim row As Integer, col As Integer, PV_MD As Variant
    Set sheet = Worksheets("Bond.Position")
    For row = 2 To sheet.UsedRange.Rows.Count
        'Fetch information
        settlementDate = sheet.Range("a" & row).Value
        coupon = sheet.Range("c" & row).Value
```

```
26          couponDate = sheet.Range("d" & row).Value
27          position = sheet.Range("e" & row).Value
28          'Calculate dirty price and modified duration
29          PV_MD = calDirtyPriceAndModifiedDuration(coupon, 2, settlementDate,
    couponDate)
30          sheet.Range("h" & row).Value = PV_MD(0)
31          sheet.Range("i" & row).Value = PV_MD(1)
32          'Calculate accrued interest
33          sheet.Range("g" & row).Value = calAccruedInterest(coupon, 2,
    settlementDate, couponDate)
34          'Clean price = dirty price - accrued interest
35          sheet.Range("f" & row).Value = sheet.Range("h" & row).Value -
    sheet.Range("g" & row).Value
36          'Calculate position basis point value
37          sheet.Range("j" & row).Value = sheet.Range("e" & row).Value * PV_MD(0) /
    100 * PV_MD(1) / 100
38      Next
39
40      'Plot Position.BPV.Curve
41      plotPositionBPV sheet, Worksheets("Position.BPV.Curve")
42  End Sub
43  '@Brief: search the range for xValues and yValues for line chart
44  Private Sub searchXYRange(sheet As Worksheet, ByRef xDictionary As Variant,
    ByRef yDictionary As Variant)
45      Dim i As Integer, n As Integer, name As String, item
46      n = sheet.UsedRange.Rows.Count
47      For i = 2 To n
48          'B: name A:XValue J:yValue
49          name = sheet.Range("b" & i).Value
50          xDictionary(name) = xDictionary(name) & "a" & i & ","
51          yDictionary(name) = yDictionary(name) & "j" & i & ","
52          'MsgBox "Name: " & name & Chr(10) & "X: " & xDictionary(name) & Chr(10)
    & "Y: " & yDictionary(name)
53      Next
54      For Each item In xDictionary.keys
55          xDictionary(item) = Left(xDictionary(item), Len(xDictionary(item)) - 1)
56          yDictionary(item) = Left(yDictionary(item), Len(yDictionary(item)) - 1)
57      Next
58  End Sub
59  '@Brief: plot Position.BPV.Curve
60  Private Sub plotPositionBPV(sheet As Worksheet, outSheet As Worksheet)
61      Dim chart As ChartObject
62      Application.ScreenUpdating = False
63      'Clear the old charts
64      If outSheet.ChartObjects.Count > 0 Then
65          outSheet.ChartObjects.Delete
66      End If
67      'Add a chart at [a1]
68      Set chart = outSheet.ChartObjects.Add(sheet.[a1].Left, sheet.[a1].Top, 400,
    300)
69      With chart.chart
70          'Config the chart
71          .HasTitle = True
72          .ChartTitle.Text = "Position.BPV.Curve"
73          .ChartTitle.Font.Size = 18
74          .HasLegend = True
75          .Legend.Font.Size = 8
76          .Legend.Font.ColorIndex = 5
```

```
77          .Legend.position = xlLegendPositionRight
78          'Config the Y label
79          With .Axes(xlValue, xlPrimary)
80              .CrossesAt = .MinimumScale
81              .TickLabels.Font.Size = 8
82              .HasTitle = True
83              .AxisTitle.Text = "Position.BPV"
84              .AxisTitle.Characters.Font.Size = 12
85              .AxisTitle.Orientation = xlUpward
86          End With
87          'Config the X label
88          With .Axes(xlCategory)
89              .TickLabels.Font.Size = 8
90              .TickLabels.Orientation = 50
91              .HasTitle = True
92              .AxisTitle.Text = "Settlement Date"
93              .AxisTitle.Characters.Font.Size = 12
94          End With
95          Dim xDictionary, yDictionary, item
96          Set xDictionary = CreateObject("Scripting.Dictionary")
97          Set yDictionary = CreateObject("Scripting.Dictionary")
98          searchXYRange sheet, xDictionary, yDictionary
99          For Each item In xDictionary.keys
100             'MsgBox "Name: " & item & " xValue: " & xDictionary(item) & "
    yValue: " & yDictionary(item)
101             With .SeriesCollection.NewSeries
102                 .Values = sheet.Range(yDictionary(item))
103                 .XValues = sheet.Range(xDictionary(item))
104                 .ChartType = xlLineMarkers
105                 .name = item
106                 .MarkerSize = 5
107                 .Format.Line.Weight = 1
108                 .MarkerStyle = xlMarkerStyleTriangle
109                 .MarkerForegroundColor = RGB(171, 130, 255)
110                 .MarkerBackgroundColor = RGB(171, 130, 255)
111                 .HasDataLabels = False
112             End With
113         Next
114     End With
115 End ub
```