SOCIETY OF ACTUARIES

Article from:

CompAct

April 2008 – Issue 27

Calculation of Generalized IRR in Excel

by Tim Rozar

Decisions about whether to proceed with a project or new product often come down to analyzing the rate of return on the project. Normally, this is a straightforward exercise involving an initial investment which is repaid over time with a stream of future positive cashflows. The discount rate that leads to a zero present value is the rate of return from the project. The calculation of this Internal Rate of Return (IRR) actually involves some tricky mathematics or the implementation of iterative numerical methods. Luckily, technology has provided tools on our desktop to easily perform this analysis. Microsoft Excel provides the IRR function, which will solve for the rate of return for a series of periodic cashflows. The basic function takes two arguments: a range of cash flows, and an initial guess. For example, assume the following investment opportunity:

Tim Rozar is vice president and actuary with RGA Reinsurance Co. He can be contacted at trozar@rgare.com

Table 1
Life Insurance Products and Finance, Atkinson & Dallas, 2000 Example 11.6.1

	C	D	E
	t	Profit(t)	NPV(t-1) at 5%
5	1	-1000	0
6	2	50	1050
7	3	50	1050
8	4	1050	1050
IRR = 5%			
In Excel: IRR(D5:D8,0.1) = 5.00%			

Unfortunately, the analysis is not always so simple. Sometimes an investment opportunity involves cumulative negative cashflows in the future. In the case where there are multiple sign changes in the projected cumulative cash flow stream, there will also exist a multiple number of real roots (IRR's) that will force the present value of the investment to zero. In such a situation, accumulated negative future cashflows may be viewed as amounts which will require additional financing beyond the returns supplied by the project.

Atkinson & Dallas suggest the Generalized ROI approach for this analysis. This approach was initially outlined by David Becker in "A Generalized Profits Released Model for the Measurement of Return on Investment for Life Insurance," (TSA 1988 Volume 40 part1 *http://www.soa.org/library/research/transactions-of-society-of-actuaries/1988/january/tsa88v40pt15.pdf)* and is therefore often referred to as the Becker IRR. Starting with the final cash flow and working backwards, a present value is calculated using the IRR as the discount rate when the present value at that duration is positive and a rate of borrowing as the discount rate when the present value is negative.

The following examples illustrate this situation:

Table 2
Life Insurance Products and Finance, Atkinson & Dallas, 2000
Example 11.6.4

	C	D	E	F	G
	T	Profit(t)	PV(t-1) if PV(t) <0 (at 7%)	PV(t-1) if PV(t) >0 (at 6.8324%)	PV(t-1)
16	1	-45		0.000	0.000
17	2	140	48.075		48.075
18	3	-55	-98.360		-98.360
19	4	-140		-46.395	-46.395
20	5	100			100.000
Traditional IRR = 11.11% or 100% (or 0%)					
In Excel: IRR(D16:D20,0.1) = 11.11% IRR(D16:D20,0.5) = 100.00%					
Generalized (Becker) IRR at 7% financing rate = 6.8324%					

Table 3
TSA 1988 Vol. 40 part1, Becker, 1988 Table 14

	C	D	E	F	G
	t	Profit(t)	PV(t-1) if PV(t) <0 (at 7%)	PV(t-1) if PV(t) >0 (at 26.271%)	PV(t-1)
30	1	50	0.000		0.000
31	2	-200		-53.500	-53.500
32	3	20		184.987	184.987
33	4	40		208.331	208.331
34	5	200		212.553	212.553
35	6	100	15.850		15.850
36	7	-70	-90.040		-90.040
37	8	-100		-21.443	-21.443
38	9	20		99.195	99.195
39	10	100			100.000
Traditional IRR = 32.61% or 275.34%					
In Excel: IRR(D30:D39,0.1) = 32.61% IRR(D30:D39,2) = 275.34%					
Generalized (Becker) IRR at 7% financing rate = 26.271%					

(continued on page 20)

The generalized or Becker IRR can be incorporated into Excel by setting up the generalized present values with IF() statements and goal-seeking for an IRR to set the present value to zero. To my knowledge, there is not an elegant way to directly calculate this metric from the cashflow stream as there is with the simple IRR(). To that end, I have developed a custom Excel VB function that will allow you to incorporate Becker IRR calculations into your spreadsheets. An important caveat should be observed, however: I'm an actuary—not a programmer. As such, the following code is undoubtedly inelegant. I encourage anyone who has developed more elegant methods for dealing with the multiple root situation in Excel to forward their suggestions to build upon this article.

The formula is set up in two steps. The BeckerOBT function calculates the Outstanding Balance (using Mr. Becker's terminology) accumulated based on either IRR borrowing rate. The BeckerIRR function then performs an iterative binary search to calculate the Generalized IRR.

The parameters needed for implementation of the BeckerIRR function are as follows:

1) EarningsRange: This is the Excel range that contains the cashflows being analyzed.
2) IntDisc: This is the discount rate to be used for financing negative cumulative cash flows.
3) BeckerIRRGuess: This is the starting point guess for the iterative search.
4) ToDecimals: This is the number of decimal places of precision for the Becker IRR result.

The code for these two functions is shown at the end of this article. A sample workbook with this function and the examples above can be e-mailed to you if you contact the editor (*Howard@Callif.org*).

This code can be inserted into each workbook that it is to be used in or it can be referenced from a personal macro workbook. To insert the code into your existing spreadsheet, choose Tools|Macro|Visual Basic Editor. From there, you may insert a new visual basic module by choosing Insert|Module. The following text can be copied and pasted into this new module.

Returning to the examples in Tables 2 and 3 above, we can now use the BeckerIRR function to directly calculate the generalized IRRs:

- Table 2: BeckerIRR(D16:D20,0.07,0.1,6) = 6.8324 percent

- Table 3: BeckerIRR(D30:D39,0.07,0.1,6) = 26.271 percent

A few notes should be observed before utilizing this function:

- Your Excel workbook will need to have macros enabled in order to use this function. This means that macro security (Tools|Macro|Security) must be set no higher than "Medium" and that macros must be enabled when prompted upon opening a worksheet using this function.
- As with all custom functions, use of this function will undoubtedly slow down calculation speed in your spreadsheet. You may wish to "comment out" the function when you don't need to refer to it.
- You may wish to reference the function from a personal macro workbook as personal.xls!BeckerIRR(). This will avoid the need to add the function to each workbook, but will make the file less portable to other users.

```
Function BeckerIRR(EarningsRange As Range, IntDisc As Double, BeckerIRRGuess
As Double, ToDecimals As Integer)
    Application.Volatile

    Dim myRange As Range
    Dim IRRa#, IRRb#, Precision#, BeckerIRRTemp#, OBt#, InitIncrement#
    Dim MaxIter%: MaxIter = 50
    Dim i%: i = 0
    InitIncrement = 0.05
    Dim ErrMsg$: ErrMsg = "Max Iter"

    BeckerIRRTemp = BeckerIRRGuess
    Precision = 10 ^ (-ToDecimals)

    OBt = BeckerOBt(EarningsRange, IntDisc, BeckerIRRGuess)

      If OBt < 0 Then
          IRRa = BeckerIRRGuess
          IRRb = IRRa
          i = 0
          Do While OBt < 0 And i < MaxIter
              IRRb = IRRb - InitIncrement
              OBt = BeckerOBt(EarningsRange, IntDisc, IRRb)
              i = i + 1
          Loop
          If i = MaxIter Then
              BeckerIRR = ErrMsg
              Exit Function
          End If
       ElseIf OBt > 0 Then
          IRRb = BeckerIRRGuess
          IRRa = IRRb
          i = 0
          Do While OBt > 0 And i < MaxIter
              IRRa = IRRa + InitIncrement
              OBt = BeckerOBt(EarningsRange, IntDisc, IRRa)
              i = i + 1
          Loop
          If i = MaxIter Then
              BeckerIRR = ErrMsg
              Exit Function
          End If
       End If
```

(continued on page 22)

```
        i = 0
        Do While Abs(IRRa - IRRb) > Precision And i < MaxIter
            BeckerIRRTemp = (IRRa + IRRb) / 2
            OBt = BeckerOBt(EarningsRange, IntDisc, BeckerIRRTemp)
            If OBt < 0 Then
                IRRa = BeckerIRRTemp
            Else
                IRRb = BeckerIRRTemp
            End If
            i = i + 1
        Loop
            If i = MaxIter Then
                BeckerIRR = ErrMsg
                Exit Function
            End If

    BeckerIRR = BeckerIRRTemp

End Function
```

Function BeckerOBt(ParamEarningsRange As Range, ParamDiscRate As Double, ParamBeckerIRR As Double)

```
    Dim myRange As Range
    Dim OBt#: OBt = 0
    Dim i%

    For Each myRange In ParamEarningsRange
        If OBt < 0 Then
            OBt = OBt * (1 + ParamBeckerIRR) + myRange.Value
        Else
            OBt = OBt * (1 + ParamDiscRate) + myRange.Value
        End If
    Next myRange

    BeckerOBt = OBt
End Function
```