

# Non-linear Yield Curve Fitting

Bjørn Eraker

Wisconsin School of Business

September 11, 2008

# Agenda for today

- Introduce curve fitting
- Demonstrate curve fitting through code written in Excel and Visual Basic

What I expect from YOU:

- 1 Basic principles of curve fitting
- 2 How to do curve fitting using the excel programs provided

We will use the curve fitting program to do cheap/rich trading later.

We will also satisfy our inner geek by looking under the hood as some VB code....

- It is not reasonable to assume that the discount function can be derived with zero error using the technique from last lecture
- One typically also has limited data on strips
- Practice on the street therefore, is often to try to extract a zero coupon yield curve/ discount function that is *implicit* in prices of various coupon bonds

Suppose it was the case that the discount function could be described by the cubic function

$$d(t) = 1 + at + bt^2 + ct^3 \quad (1)$$

where

- the argument  $t$  is the *maturity*, and
- $a, b, c$  are *parameters* that we can choose in whichever way we want

The cubic equation is just one possible function. You can choose whichever function you want, essentially..

A popular function to use for yield curve estimation is the so-called Nelson-Siegel function. This function has four shape parameters  $b_1$ ,  $b_2$ ,  $b_3$  and  $\lambda$ :

$$y(t) = b_1 + b_2 \frac{1 - e^{-\lambda t}}{\lambda t} + b_3 \left( \frac{1 - e^{-\lambda t}}{\lambda t} - e^{-\lambda t} \right) \quad (2)$$

If (??) is a reasonable functional form, we might proceed by assigning numerical values to the parameters  $a$ ,  $b$ ,  $c$  in some reasonable way.

We should choose  $a$ ,  $b$ ,  $c$  such that the discount function  $d(t)$  prices the existing coupon bonds as best as possible.

In other words, we want to choose  $a$ ,  $b$ ,  $c$  such that  $d(t)$  **minimizes the pricing errors.**

# Pricing errors

Let  $P_i$  denote the current price of bond  $i$ . Let  $P_i(\Theta)$  denote the *theoretical price*. We let  $\Theta$  denote a set of unknown parameters such as  $a, b, c$  in the cubic example.

The theoretical price is obtained as the present value

$$P_i(\Theta) = \sum_{i=1}^n d(t_i, \Theta) C(t_i)$$

where  $d(t, \Theta)$  now denotes the discount function's dependence on the unknown parameters,  $\Theta$ .

The *pricing errors* are defined as the difference between the market and model prices

$$\text{pricing error } i = P_i - P_i(\Theta)$$

# Minimizing Pricing Errors

We would like our discount function to do "as good of a job as possible."

$$\sum_{i=1}^N |P_i - P_i(\Theta)| \quad (3)$$

or the squared errors

$$\sum_{i=1}^N (P_i - P_i(\Theta))^2 \quad (4)$$

# Implementation in Excel

We will show how to do the curve fitting in Excel. The program uses the following tricks

- The market prices,  $P_i$  are imported from Bloomberg and placed in some column in the spreadsheet
- Some range of cells in the spreadsheet needs to contain the parameters of interest ( $a, b, c$ )
- We need a compact way to compute the theoretical value of the coupon bonds.
- Here: discount function/ theoretical bond prices computed through function written in Visual Basic

# Computing theoretical prices

We compute the prices using Visual Basic functions.

The function needs the following inputs:

- 1 The settlement date
- 2 Maturity date
- 3 Coupon rate
- 4 The parameters  $\Theta$  that determine the shape of the discount function

# Using Visual Basic with Excel

- VB is Microsoft's proprietary programming language. It is loosely based on "old" basic - a very simple language
- VB comes with excel but is unavailable on Excel for mac
- Open the file "discountfunctions.xls"
- You need to click "enable macros" when opening the course files that use VB
- From the "Developer" menu, click on the "Visual Basic" tab.

## Code for Nelson-Siegel

```
Function NelsonSiegel(timeToExp, b1, b2, b3, lambda)
    temp = Exp(-lambda * timeToExp)
    NelsonSiegel = b1 + b2 * (1 - temp) / (lambda * timeToExp)
                + b3 * ((1 - temp) / (lambda * timeToExp) - temp)
End Function
```

## Comments:

- First line tells the computer that we are declaring a function with name NelsonSiegel and arguments timeToExp, b1, ..., lambda.
- You can now call 'NelsonSiegel' like any other function in your spreadsheet. For example

```
=NelsonSiegel (A1, A2, A3, A4, A5)
```

in some cell evaluates the function at arguments in cells A1 through A5.

We use the function 'NSbondPrice' to compute bond prices using the Nelson Siegel discount function. The function takes as argument

- the settlement date
- maturity
- coupon
- parameters  $b_1, b_2, b_3$  and  $\lambda$  that determine the shape of the discount function

```
Function NSbondPrice(settlement,  
                    expiration, coupon, b1, b2, b3, lambda)
```

# Calibrating Parameters

We want to choose parameters such that the pricing errors are as small as possible.

In the spreadsheet, we put absolute pricing errors in cells F4 and G4 for the Nelson Siegel.

The parameters are placed in cells E4-E7.

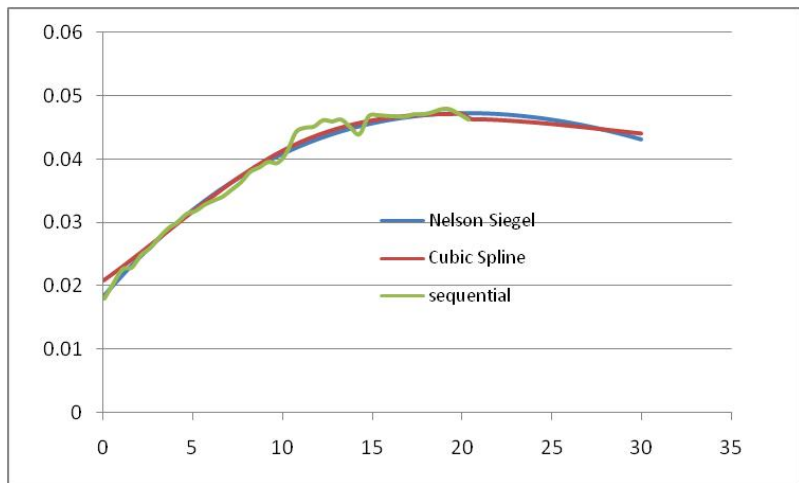
We can now use the *built in* solver tool to find the values of parameters  $b_1 - \lambda$  that minimize the absolute pricing errors. The way we do this is to specify F8 as the target cell, click the "min" button (for minimize), and specify cells E4-E7 to be changed.

# "Brute force" Excel computation of a bond price

```
Basic "Brut Force" bond Price Computation
maturity 8/15/2011 5
coupon 5
today 7/15/2008
NELS. SIEG. CUBIC SPLINE
time t CF r_ns(t) present value r_cs(t) present value
8/15/2008 0 0.084931507 2.5 0.018678752 2.496037108 0.020966979 2.495552069
2/15/2009 1 0.591780822 2.5 0.020307694 2.470135551 0.022012377 2.467644938
8/15/2009 2 1.090410959 2.5 0.021852132 2.44113459 0.023067516 2.437901576
2/15/2010 3 1.589041096 2.5 0.023340077 2.408977515 0.024144586 2.405899851
8/15/2010 4 2.087671233 2.5 0.024772622 2.373993697 0.025239067 2.371683067
2/15/2011 5 2.589041096 2.5 0.026158266 2.336293984 0.02635255 2.335119104
8/15/2011 6 3.084931507 102.5 0.027475792 94.17001899 0.027462179 94.17397387

108.6965914 108.6877745
```

Explanation: These computations are included to demonstrate that for this particular bond, the value is the same as when using the Visual Basic valuation functions (as in line 106 in valuation sheets)



**Figure:** Zero yld curves computed with recursive method from FEB/AUG maturities and Nelson-Siegel and cubic spline curve fitting methods.

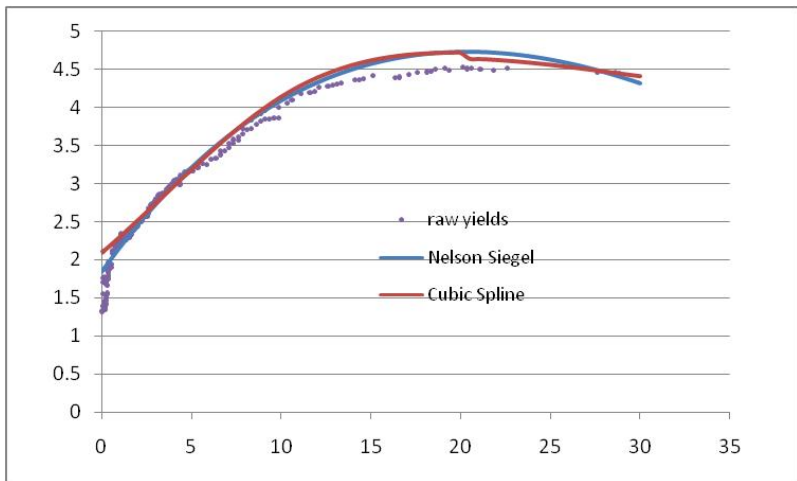


Figure: Zero yld curves plotted alongside the raw coupon yields.

- NS and CS are both smoother than sequential
- Could still use smoother functional forms...
- Note that most coupon ytm's are below the zero curve. Take a 20 year coupon bond as an example. This bond is essentially a portfolio of short bonds with portfolio weights proportional to the size of the coupons. The ytm on this coupon bond is a weighted average of ytm's of all zeros with maturities 6 mos-20 yrs. As such, the coupon ytm will lie below the 20 year zero coupon ytm

# Pricing Errors

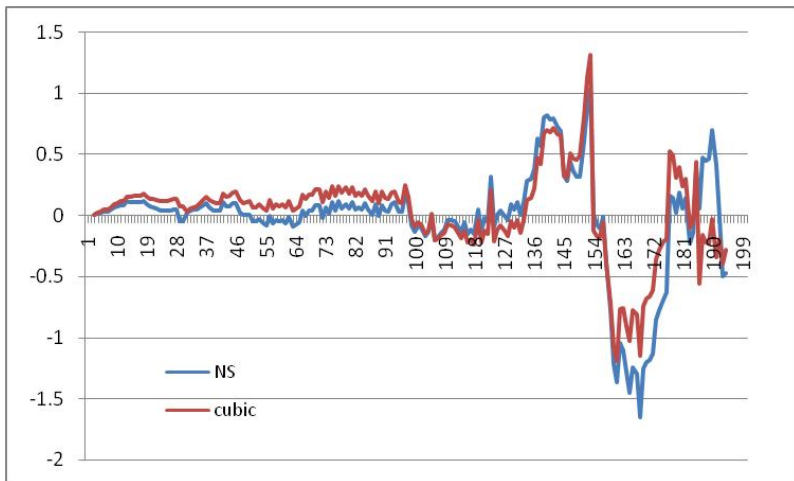


Figure: Pricing errors for NS and CS curve fitting.

# Curvefitting - Concluding Remarks

- How much to smooth? More of an art than science:
- Overfitting will give unreliable results. We may not see profit opportunities and we may adapt potential mispricing in our own pricing decisions
- Underfitting may cause incorrect pricing as well
- Pricing errors on the order of 100 and 150 basis points may be too much
- Which method is better... stay tuned for homework.