Non-linear Yield Curve Fitting

Bjørn Eraker

Wisconsin School of Business

December 27, 2014

Agenda for today

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

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.
Full VB code explanation.

Parameterizing a discount function

Consider again the present value formula

$$V_0 = \sum_{i=1}^{n} C(t_i) d(t_i). \tag{1}$$

Let's now assume that $d()$ is some undefined function. We want it to be a parametric function.

$d(t)$ should have the following properties:

- $0 < d(t) < 1$ - the value of a dollar paid in the future is positive but less than a dollar today
- $d(t) < d(s)$ for $t > s$, - sooner is better
- $\lim_{t \to \infty} d(t) = 0$ - the value of a dollar paid infinitely far into the future is zero

Two candidate functions

Consider

$$d(t) = (1 + y(t))^{-t} \tag{2}$$

and

$$d(t) = \exp(-y(t)t) \tag{3}$$

to be candidate functions. These functions are the prices of zero coupon bonds with maturity t if the argument $y(t)$ (another function) is interpreted as the yield-to-maturity of the zero. These functions allow us to map the YTM of a zero coupon bond into a discount function.

Zero coupon yield curve functions: Nelson-Siegel

So we need some function $y(t)$ to represent the zero coupon yield curve. We will study the so-called Nelson-Siegel function, although the set of choices is nearly unlimited....
The NS function has four shape parameters b_1, b_2, b_3 and λ:

$$y(t) = b_1 + b_2 \frac{1 - e^{-\lambda t}}{\lambda t} + b_3 \left(\frac{1 - e^{-\lambda t}}{\lambda t} - e^{-\lambda t} \right) \quad (4)$$

Example

Let's use the Nelson-Siegel zero coupon function along with (2) or (3) to price a bond. Assume the parameter values

$$b_1 = 0.001 \tag{5}$$
$$b_2 = 0.027707887 \tag{6}$$
$$b_3 = 0.232637296 \tag{7}$$
$$\lambda = 0.077145317 \tag{8}$$

for the NS function.
We will price a bond with a 8/15/2013 maturity, Pretend today is 1/29/2010. The bond has 4.25% coupon.

We start by computing $y(t)$, the zero coupon yield using Nelson-Siegel. The first coupon payment has 0.04657 years to maturity. We get $y(0.04657) \approx 0.029$.

The first coupon payment is therefore worth

$$\exp(-0.04657 \times 0.029) \times 2.125 = 2.1221$$

and so on.

The spreadsheet on the next slide computes the present value and gives the theoretical bond price.

					NELS. SIEG.	
time		t	CF	y(t)	present val	r
2/15/10	0	0.0465753	2.125	0.029075	2.122124	
8/15/10	1	0.5452055	2.125	0.032891	2.087234	
2/15/11	2	1.0465753	2.125	0.03652	2.045314	
8/15/11	3	1.5424658	2.125	0.039914	1.99812	
2/15/12	4	2.0465753	2.125	0.043173	1.945298	
8/15/12	5	2.5452055	2.125	0.046216	1.88918	
2/15/13	6	3.0493151	2.125	0.049118	1.829416	
8/15/13	7	3.5452055	102.125	0.051808	84.98947	
					98.90615	

Figure: Excel Screen Shoot.

VB Computation

We can also compute the theoretical bond price by writing a VB function that takes as input the various characteristics of the bond. The function 'NSbondPrice' does this.

It is called as

NSbondPrice(settle, expiration, cpn, b1, b2, b3, lambda)

where the first two arguments are the settlement date and expiration date. The third is the coupon (in percent). The last three are the parameters in the NS function.

By calling ’NSbondPrice’ in any cell in the spreadsheet (using the same values as above) we get 98.906 - the same as with the ”manual method."

Explaining NSbondPrice

The function is

```
Function NSbondPrice(settle, expiration, cpn, b1, b2, b3, lambda)

    n = WorksheetFunction.CoupNum(settle, expiration, 2)
    firstCPNd = WorksheetFunction.CoupNcd(settle, expiration, 2)

    t = 365
    s = (firstCPNd - settle) / t
    yt = NelsonSiegel(s, b1, b2, b3, lambda)
    bondValue = cpn * 50 * Exp(-s * yt)
    i = 1
    Do While i < n
        nextCPNd = nCPNdate(firstCPNd, i)
        s = (nextCPNd - settle) / t
        bondValue = bondValue + cpn * 50 * Exp(-s * NelsonSiegel(s, b1, b2, b3, lambda))
        i = i + 1
    Loop
    NSbondPrice = bondValue + 100 * Exp(-s * NelsonSiegel(s, b1, b2, b3, lambda))

End Function
```

The computations go as follows: (note that all dates are integers using the internal excel date numbering)

- Line 2: The variable n is defined by calling the built-in excel function 'CoupNum'. This function returns the number of coupon payments as a function of the settle date and expiration date, and number of annual coupon payments.
- Line 3: The variable 'firstCPNd' is defined by calling the built-in Excel function 'Coupncd' which gives the first date the bond pays coupon.
- Line 6: s is defined to be the time to the first coupon payment in years (ie. the number of days to the first coupon / 365

- Line 7: *yt* is defined as the first point on the zero coupon curve as a function of *s* computed in Line 6.
- Line 8: The variable 'bondValue' is initialized. It is the annual coupon (percentage), multiplied by 100 / 2= 50, which is again multiplied by the discount function $\exp(-s \times yt)$ where *s* and *yt* were computed in lines 6 and 7.

- Lines 10-15. Here we are looping. For each iteration of the loop we are adding the present value of the future coupon payments to the present value of the bond. The results of this is to compute the sum over the present values of the future coupon payments. The statement

```
bondValue = bondValue + cpn * 50 * Exp(-s * NelsonSiegel(s, b1, b2, b3, lambda))
```

  accomplishes this.
- Line 11: This ads the present value of the principal ($100) to the present value of the coupon payments and returns the result.

Why VB?

- The only reason to use VB is that we can compute the theoretical value of the bond (as computed with the 'NSbondPrice' function) in a single cell in the spreadsheet. We do not need a new sheet for every single treasury bond we are interested in.
- So suppose we now compare the market bond price (in real time) to the theoretical bond price in two columns of a spreadsheet....

(see next page)

E4 f_x -0.03

	A	B	C	D	E	F	G
1	Mkt Price	Th. Price	Pricing Erro	abs(diff)	parameters	sq. er w/o bills	sq err w bills
2							
3							
4	101.0764266	101.0637453	0.012681	0.0126813	-0.03	209.8710789	210.3674
5	101.7663043	101.7591225	0.007182	0.0071818	0.027708		
6	102.392663	102.3841786	0.008484	0.0084845	0.232637		
7	103.2601902	103.259257	0.000933	0.0009332	0.077145		
8	101.0162293	101.0136595	0.00257	0.0025698			
9	102.0073377	102.0166528	-0.00932	0.0093151			
10	100.8894231	100.8915754	-0.00215	0.0021523			
11	101.9975962	102.0141226	-0.01653	0.0165264			
12	101.0693198	101.0712722	-0.00195	0.0019525			
13	101.9327521	101.9384711	-0.00572	0.005719			
14	102.2323895	102.2509741	-0.01858	0.0185846			
15	101.3004808	101.3022744	-0.00179	0.0017936			
16	101.7819368	101.7890247	-0.00709	0.0070879			
17	101.3942507	101.3984187	-0.00417	0.004168			
18	101.8891575	101.8799205	0.009237	0.009237			
19	104.0326087	104.0200367	0.012572	0.012572			

Figure: Excel Screen Shoot.

Curve Fitting

Note that the theoretical prices on the previous example excel screen are 'close' but not identical to the observed market prices. For example, the first bond differs by 1.2 cents, ...

We can play around with the parameters $b1, b2, b3$ and λ to get different theoretical prices.

Pricing errors

Let P_i denote the current price of bond i. Let $P_i(\Theta)$ denote the *theoretical price*. We let Θ denote a set of unknown parameters such as $b1, b2, b3, \lambda$.
The *pricing errors* are defined as the difference between the market and model prices

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

We can choose $b1, b2, b3, \lambda$ such that we minimize the pricing errors. Define

$$\frac{1}{N}\sum_{i=1}^{N}|P_i - P_i(\Theta)| \tag{9}$$

to be the average absolute error, or

$$\frac{1}{N}\sum_{i=1}^{N}(P_i - P_i(\Theta))^2 \tag{10}$$

to be the average squared error.

Minimizing Pricing Errors

We can now minimize the average absolute pricing error or the average squared error.

How?

Use solver in Excel!

Why do curvefitting?

- It gives us a theoretical zero coupon bond curve that is extracted from coupon prices
- Pricing errors may suggest trading opportunities
- Note: Nelson-Siegel is very restrictive. Wall-St typically use more flexible functions.

Next time: A different zero coupon curve!