CHAPTER 1

SWAPS AND FIXED INCOME INSTRUMENTS

SECTIONS

Swaps are often used to hedge interest rate risk exposure to balance sheets as well as for bond and loan portfolios. By matching the durations of balance sheet fixed income assets and liabilities (e.g., bond or loan instruments), swaps can immunize the balance sheet from interest rate risk. Ideally, the hedge should match both the duration and timing of cash flows of the fixed income portfolio with that of the swap as closely as possible. For instance, if a bank has a portfolio of floating-rate loans that have a duration of five years, the bank can enter a swap with a duration of five years (the duration of the swap is the difference between the fixed- and floating-leg durations of the swap) to receive the fixed rate and to pay floating (effectively changing the characterization of the loan from floating to fixed and thereby locking in a fixed rate of return.) Although in general, basis risk exists because there is not an exact match between the cash flows—e.g., interest rate payments on the loans and those of the swap do not match—the bank has reduced its exposure to shifts in the yield curve. Moreover, institutional money managers can also use a combination

of Chicago Board of Trade (CBOT) Treasury note futures and swap futures to structure hedges to protect a portfolio of corporate and Treasury securities from a rise in interest rates. Due to the liquidity and standardization of swap futures, swap futures are becoming a cheaper and more efficient way to hedge a fixed income portfolio than entering a swap.[1] This chapter discusses the details of hedging interest rate risk and bond portfolios using swaps and fixed income instruments (e.g., futures).

In §1.1, we discuss using Eurodollar futures to compute LIBOR swap rates. In §1.2, we discuss Treasury bills and Treasury bonds, including how they are quoted and priced. In §1.3, we discuss bootstrapping the yield curve to compute discount swap rates. In §1.4, we discuss hedging debt positions and interest rate movements using fixed income instruments. In §1.5, we discuss bond duration, modified duration, and DV01 (dollar value of a one-basis-point move) calculations, which are necessary for computing swap durations and modified durations because a fixed-for-floating interest rate swap is composed of two legs that are equivalent to a fixed-rate bond and a floating-rate bond. In §1.6, we discuss term structure of rates, and in §1.7, we discuss how to numerically bootstrap the yield curve. In §1.8, we discuss bootstrapping in Matlab and provide examples, while in §1.9, we discuss bootstrapping using Excel. In §1.10, we discuss general swap pricing in Matlab using the Black-Derman-Toy (BDT) and Heath-Jarrow-Morton (HJM) interest rate models, and in §1.11, we discuss swap pricing using term structures like the forward curve built from zero-coupon and coupon-bearing bond cash flows. In §1.12, we implement and price fixed-for-floating swaps, including calculations for duration and risk measures, using C++. Finally, in §1.13, a Bermudan swaption pricing implementation in Matlab is provided.

1.1 EURODOLLAR FUTURES

Eurodollar futures[2] contracts maturing in March, June, September, and December are sometimes used to calculate the LIBOR swap zero rates for swap maturities greater than one year. The Eurodollar futures interest rate can be used to compute forward rates for long-dated maturities. In the United States, spot LIBOR rates are usually used to define the LIBOR zero curve for maturities up to one year. Eurodollar futures are typically then used for maturities between one and two years and sometimes for maturities up to five, seven, and ten years. In addition, swap rates, which define par yield bonds, are used to calculate the zero curve for maturities longer than a year. Using a combination of spot LIBOR rates, Eurodollar futures, and swap rates, the LIBOR/swap zero curve can be generated using a bootstrap method procedure.

Typically, a convexity adjustment is made to convert Eurodollar futures rates into forward interest rates. For short maturities (up to one year), the Eurodollar futures interest rate can be assumed to be the same as the corresponding forward interest rate. But for longer maturities, the difference between futures and forward contracts becomes important when interest rates vary unpredictably.[3]

Suppose the quoted Eurodollar futures price is P; then the cash contract price is

$$10000\,(100 - 0.25(100 - P)) \tag{1.1}$$

which is equivalent to 10,000 times the cash futures price of $100 - 0.25(100 - P)$. Suppose $P = 96.7$; then the contract price is

$$10000\,(100 - 0.25(100 - 96.7)) = \$991,750$$

A Eurodollar contract is like a Treasury bill contract, but with some important differences. Both the Eurodollar and T-bill contracts have an underlying face value amount of $1 million. However, for a T-bill, the contract price converges at maturity to the price of a 91-day $1 million face-value Treasury bill, and if the contract held to maturity, this is the instrument delivered.[4] A Eurodollar futures contract is cash settled on the second London business day before the third Wednesday of the month.[5] The final marking to market sets the contract price equal to

$$f_0 = (\$1,000,000)\left(\frac{100 - 0.25R}{100}\right) = 10,000(100 - 0.25R)$$

where R is the quoted LIBOR Eurodollar rate at that time.[6] This quoted Eurodollar rate is the actual 90-day rate on Eurodollar deposits with quarterly compounding.[7] It is not a discount rate. As Hull states, "the Eurodollar futures contract is therefore a futures contract on an interest rate, whereas the Treasury bill futures contract is a futures contract on the price of a Treasury bill or a discount rate."[8]

1.2 Treasury Bills and Bonds

Treasury bills are short-term discount securities issued by the U.S. Treasury. At the time of sale, a percentage discount is applied to the face value. Treasury bill prices are quotes as a discount rate on a face value of $100. At maturity, the holder redeems the bill for full face value. The basis (or day counting convention) for the interest accrual is actual/360 so that interest accrues on the actual number of elapsed days between purchase and maturity, assuming each year has 360 days. The Treasury price quote is the annualized dollar return provided by the Treasury bill in 360 days expressed as a percentage of the face value

$$\frac{360}{n}(100 - P) \tag{1.2}$$

where P is the cash price of a Treasury bill that has a face value of $100 and n days to maturity.[9] The discount rate is not the rate of return earned on the Treasury bill. If the cash price of a 90-day Treasury bill is 99, the quoted price would be 1.00. The *rate of return* would be 1/99, or 1.01%, per 90 days. This is translates to

$$\frac{1}{99} \times \frac{360}{90} = 0.0404$$

or 4.04% per annum on an actual/360 basis. Alternatively, it is

$$\frac{1}{99} \times \frac{365}{90} = 0.04096$$

or approximately 4.10% per annum on an actual/365 basis. Both of these rates are expressed with a quarterly compounding period of 90 days. In order to directly compare Treasury yields with yields quoted on Treasury bonds, often semiannual compounding (a

compounding period of 180 days) is used on an actual/365 basis. The computed rate is known as the *bond equivalent yield*. In this case, the bond equivalent yield is

$$\frac{1}{99} \times \frac{365}{180} = 0.02048.$$

For short-term Treasury bills (fewer than 182 days to maturity), the *money-market yield* can be computed as 360/365 of the bond equivalent yield. In this example, it is 2.02%.

T-bill futures call for the delivery or purchase of a T-bill with a maturity of 91 days and a face value of $1,000,000. They are used for speculating and hedging short-term rates. Prices of T-bill futures are quoted in terms of the interbank money market (IMM) index or discount yield R_d:

$$\text{IMM} = 100 - R_d$$

Theoretical T-bill pricing is done with a carrying cost model

$$f_0 = S_0^M (1 + R_f)^T \tag{1.3}$$

where

$$\begin{aligned} f_0 &= \text{price of T-bill futures} \\ T &= \text{time to expiration on futures} \\ S_0^M &= \text{price on spot T-bill with maturity of } M = 91 + T \\ R_f &= \text{risk-free or repo rate} \end{aligned}$$

Following Johnson (2004),[10] suppose the rate on a 161-day spot T-bill is 5.7% and the repo rate (or risk-free rate) for 70 days is 6.38%; then the price on a T-bill futures contract with an expiration of 70 days would be

$$f_0 = (97.5844)(1.0638)^{70/365} = 98.7487$$

where

$$S_0^{161} = \frac{100}{(1.057)^{161/365}} = 97.5844.$$

The future price is governed by arbitrage considerations. If the futures market price is above f^*, arbitrageurs would short the futures contract and go long the spot T-bill. For example, suppose the futures market price is $f^{M=70/365} = 99$. An arbitrageur would go short in the futures, agreeing to sell a 91-day T-bill at 99, 70 days later, and would go long the spot, borrowing 97.5844 at 6.38% for 70 days to finance the purchase of the 161-day T-bill that is trading at 97.5844. Seventy days later at expiration, the arbitrageur would sell the T-bill (which would now have a maturity of 91 days) on the futures for $f^{M=70/365} = 99$ and pay off his financing debt of $f^* = 98.74875$, realizing a cash flow (CF_T) of $2,513:

$$\begin{aligned} CF_T &= f_0^M - f_0^* \\ &= f_0^M - S_0^M (1 + R_f)^T \\ &= 99 - 97.5844(1.0638)^{70/365} \\ &= 99 - 98.7487 = 0.2513 \end{aligned}$$

so that the cash flow or profit is

$$CF_T = (\$1,000,000)\left(\frac{0.2513}{100}\right) = \$2,513.$$

Note that if $f^M = 99$, a money market manager planning to invest for 70 days in T-bills at 6.38% could earn a greater return by buying a 161-day bill and going short the 70-day T-bill futures to lock in the selling price. For example, using the preceding numbers, if a money manager was planning to invest 97.5844 for 70 days, she could buy a 161-day bill for that amount and go short in the futures at 99. Her return would be 7.8%, compared to only 6.38% from the 70-day T-bill:

$$R = \left(\frac{99}{97.5844}\right)^{365/70} - 1 = 0.078$$

If the market price is below f^*, then arbitrageurs would go long in the futures and short in the spot. Suppose $f^M = 98$. An arbitrageur would go long in the futures, agreeing to buy a 91-day T-bill for 98 seventy days later and would go short in the spot, borrowing the 161-day T-Bill, selling it for 97.5844 and investing the proceeds at 6.38% for 70 days. Seventy days later (expiration), the arbitrageur would buy the bill (which now would have a maturity of 91 days) on the futures for 98 (fm), use the bill to close his short position, and collect 98.74875 (f^*) from his investment, realizing a cash flow of \$7487.

$$\begin{aligned} CF_T &= f_0^* - f_0^M \\ &= S_0^M(1+R_f)^T - f_0^M \\ &= 97.5844(1.0638)^{70/365} - 98 \\ &= 98.7487 - 98 = 0.2513 = 0.7487 \end{aligned}$$

so that the cash flow or profit is

$$CF_T = (\$1,000,000)\left(\frac{0.7487}{100}\right) = \$7,487$$

If the carrying-cost model holds, then the spot rate on a 70-day bill (repo rate) will be equal to the synthetic rate (implied repo rate) formed by buying the 161-day bill and going short in the 70-day futures:

Buy 161-day T-bill $S_0^{161} = 97.5844$.

Short position in T-bill futures at $f_0^M = f_0^* = 98.74875$.

$$R = \left(\frac{98.74875}{97.5844}\right)^{365/70} - 1 = 0.0638.$$

Furthermore, if the carrying-model holds, then the yield-to-maturity, *YTM*, of the futures will be equal to the implied forward rate F. Locking in the 91-day investment to be made 70 days from now, as follows:

1. Short 70-day T-bill at $S_0^{70} = 98.821$.
2. Buy $n = \frac{S_0^{70}}{S_0^{161}} = \frac{98.821}{97.5844} = 1.01267$ of 161-day bill at 97.5844.
3. End of 70 days, cover short bill for 100.
4. 90 days later, collect on the investment in the original 161-day bill: $1.01267(100) = 101.267$.

$$R = \left(\frac{101.267}{100}\right)^{365/91} - 1 = 0.0518 = F_{91,70}$$

which is equal to

$$YTM_f = \left(\frac{100}{98.74875}\right)^{365/91} - 1 = 0.0518.$$

Hedging with T-Bill Futures

T-bill futures are often used for hedging by money managers. Suppose a money manager is expecting a \$5 million cash flow in June, which she plans to invest in a 91-day T-bill. With June T-bill futures trading at IMM of 91 (*June IMM* $= 91$ or $R_D = 9\%$), the manager could lock in a 9.56% rate by going long 5.115 June T-bill contracts:

$$f_0^{June} = (\$1,000,000)\left(\frac{100-(9)(0.25)}{100}\right) = \$977,500$$

$$YTM_f = \left(\frac{\$1,000,000}{\$977,500}\right)^{365/91} - 1 = 0.0956$$

$$n_f = \frac{CF_T}{f_0} = \frac{\$5,000,000}{\$977,500} = 5.115 \text{ long contracts}$$

Suppose in June, the spot 91-day T-bill rate is at 8%. The manager would find T-bill prices higher at \$980,995, but would realize a profit of \$17,877 from closing the futures position. Combining the profit with the \$5 million cash flow, the manager would be able to buy 5.115 T-bills[11] and earn a rate off the \$5 million investment of 9.56%:

At June contract maturity, rate on T-bill $= 8\%$.

$$\text{spot rate} = S_T^{91} = \frac{\$1,000,000}{(1.08)^{91/365}} = \$980,995.$$

$$\text{profit} = \pi_f = [\$980,995 - \$997,500](5.115) = \$17,877.$$

$$\text{hedge ratio} = n_{TB} = \frac{CF + \pi_f}{S_T^{91}} = \frac{(\$5,000,000 + \$17,877)}{\$980,995} = 5.115.$$

$$\text{contract price} = f_0^{June} = (\$1,000,000)\left(\frac{100-(9)(0.25)}{100}\right) = \$977,500.$$

$$\text{rate of return} = R = \left[\frac{(\$1,000,000)(5.115)}{\$5,000,000}\right]^{365/91} - 1 = 0.956 = 9.56\%.$$

Suppose instead that in June, the spot 91-day T-bill rate is at 10%. The manager would find T-bill prices lower at \$976,518, but would realize a loss of \$5,025 from closing the futures position. After paying the clearing house \$5,025, the manager would still be able to buy 5.115 T-bills given the lower T-bill prices, earning a rate of return from the \$5 million investment at 9.56%:

At June contract maturity, rate on T-bill $= 10\%$.

$$S_T^{91} = \frac{\$1,000,000}{(1.10)^{91/365}} = \$976,518.$$
$$\pi_f = [\$976,518 - \$997,500]\,(5.115) = -\$5,025.$$
$$n_{TB} = \frac{CF + \pi_f}{S_T^{91}} = \frac{(\$5,000,000 - \$5,025)}{\$976,518} = 5.115.$$
$$f_0^{June} = (\$1,000,000)\left(\frac{100 - (9)(0.25)}{100}\right) = \$977,500.$$
$$R = \left[\frac{(\$1,000,000)(5.115)}{\$5,000,000}\right]^{365/91} - 1 = 0.956 = 9.56\%.$$

Note that at any rate, the money market manager earns a rate of return of 9.56%.

Long Futures Hedge: Hedging Synthetic Futures on 182-Day T-Bill

Suppose a money market manager is expecting a \$5 million cash flow in June, which she plans to invest in a 182-day T-bill. Because the T-bill underlying a futures contract has a maturity of 91 days, the manager would need to go long in both June T-bill futures and a September T-bill futures (note that there are approximately 91 days between the contract) in order to lock in a return on a 182-day T-bill instrument. If June T-bill futures were trading at IMM of 91 and September futures were trading at IMM of 91.4, then the manager could lock in a 9.3% rate on an instrument in 182-day T-bills by going long in 5.115 June T-bill futures and 5.11 September contracts:

June IMM $= 91$ or $R_D = 9\%$
Sept IMM $= 91.4$ or $R_D = 8.6\%$

$$f_0^{June} = (\$1,000,000)\left(\frac{100 - (9)(0.25)}{100}\right) = \$977,500$$
$$f_0^{Sept} = (\$1,000,000)\left(\frac{100 - (8.6)(0.25)}{100}\right) = \$978,500$$
$$YTM_f^{June} = \left[\frac{\$1,000,000}{\$977,500}\right]^{365/91} - 1 = 0.0956$$
$$YTM_f^{Sept} = \left[\frac{\$1,000,000}{\$978,500}\right]^{365/91} - 1 = 0.091$$
$$n_f^{June} = \frac{CF_T}{f_0} = \frac{\$5,000,000}{\$977,500} = 5.115 \text{ long contracts}$$

$$n_f^{Sept} = \frac{CF_T}{f_0} = \frac{\$5,000,000}{\$978,500} = 5.112 \text{ long contracts}$$

so that the return is

$$YTM_f^{182} = \left[(1.0956)^{91/35}(1.091)^{91/365}\right]^{365/182} - 1 = 0.093.$$

Suppose in June, the 91-day T-bill rate is at 8% and the spot 182-day T-bill rate is at 8.25%. At these rates, the price on the 91-day spot T-bill would be

$$S_T^{91} = \frac{\$1,000,000}{(1.08)^{91/365}} = \$980,995$$

and the price on the 182-day spot T-bill would be

$$S_T^{182} = \frac{\$1,000,000}{(1.08)^{182/365}} = \$961,245.$$

If the carrying-cost model holds, then the price on the September futures at the June date is

$$f_0^{Sept} = S_0^{182}(1 + R_f)^T = \$961,245(1.08)^{91/365} = \$979,865.$$

At these prices, the manager would be able to earn futures profits of

$$\text{June } \pi_f = [\$980,995 - \$977,500]5.115 = \$17,877$$
$$\text{Sept } \pi = [\$979,865 - \$978,500]5.11 = \$6,975$$

for a total profit of $24,852 from closing both futures contract (which offsets the higher T-bill futures prices) and would be able to buy

$$n_{TB} = \frac{\$5,000,000 + \$24,852}{\$961,245} = 5.227$$

182-day T-bills, yielding a rate of return of

$$R = \left[\frac{(5.227)(\$1,000,000)}{\$5,000,000}\right]^{365/182} - 1 = 0.093$$

or 9.3% from a $5 million investment. Readers can verify for themselves that the rate of return will still be 9.3% if the 91-day and 182-day T-bill spot rates rise. Thus, Treasury bond futures contracts call for the delivery or purchase of a T-bond with a face value of $100,000. The contract allows for the delivery of a number of T-bonds; there is a conversion factor used to determine the actual price of the futures given the bond that is delivered. In actuality, the cheapest-to-deliver T-bond is delivered. T-bond futures are quoted in terms of a T-bond with an 8% coupon, semiannual payments, maturity of 15 years, and face value of $100.

In Matlab, one can make Treasury bills directly comparable to Treasury notes and bonds by restating U.S. Treasury bill market parameters in U.S. Treasury bond form as zero-coupon bonds via the following function:

```
[TBondMatrix, Settle] = tbl2bond(TBillMatrix)
```

`TBillMatrix` are the Treasury bill parameters. An N-by-5 matrix is where each row describes a Treasury bill. N is the number of Treasury bills. Columns are `[Maturity DaysMaturity Bid Asked AskYield]`, as described in Table 1.1.

Table 1.1

`Maturity`	Maturity date, as a serial date number. Use `datenum` to convert date strings to serial date numbers.
`DaysMaturity`	Days to maturity, as an integer. Days to maturity is quoted on a skip-day basis; the actual number of days from settlement to maturity is `DaysMaturity + 1`.
`Bid`	Bid bank-discount rate: the percentage discount from face value at which the bill could be bought, annualized on a simple-interest basis. A decimal fraction.
`Asked`	Asked bank-discount rate, as a decimal fraction.
`AskYield`	Asked yield: the bond-equivalent yield from holding the bill to maturity, annualized on a simple-interest basis and assuming a 365-day year. A decimal fraction.

The output consists of the Treasury bond parameters given in **`TBondMatrix`**, an N-by-5 matrix where each row describes an equivalent Treasury (zero-coupon) bond. Columns are `[CouponRate Maturity Bid Asked AskYield]`, as described in Table 1.2

Table 1.2

`CouponRate`	Coupon rate, which is always 0.
`Maturity`	Maturity date, as a serial date number. This date is the same as the Treasury bill `Maturity` date.
`Bid`	Bid price based on $100 face value.
`Asked`	Asked price based on $100 face value.
`AskYield`	Asked yield to maturity: the effective return from holding the bond to maturity, annualized on a compound-interest basis.

Example 1

Given published Treasury bill market parameters for December 22, 1997:

```
TBill = [datenum('jan 02 1998') 10 0.0526 0.0522 0.0530
         datenum('feb 05 1998') 44 0.0537 0.0533 0.0544
         datenum('mar 05 1998') 72 0.0529 0.0527 0.0540];
```

Execute the function:

```
TBond = tbl2bond(TBill)

TBond =
             0 729760 99.854 99.855 0.053
             0 729790 99.344 99.349 0.0544
             0 729820 98.942 98.946 0.054
```

1.3 Computing Treasury Bill Prices and Yields in Matlab

In Matlab, you can specify T-bills yield as money-market or bond-equivalent yield.

Matlab Treasury bill functions all assume a face value of $100 for each Treasury bill. The price of a T-bill can be computed using the function

```
Price = prtbill(Settle, Maturity, Face, Discount)
```

where the arguments are as listed in Table 1.3.

Table 1.3

`Settle`	Enter as serial date number or date string. `Settle` must be earlier than or equal to `Maturity`.
`Maturity`	Enter as serial date number or date string.
`Face`	Redemption (par, face) value.
`Discount`	Discount rate of the Treasury bill. Enter as decimal fraction.

Example 2

```
Settle = '2/10/2005';
Maturity = '8/7/2005';
Face = 1000;
Discount = 0.0379;
Price = prtbill(Settle, Maturity, Face, Discount);

Price =
      981.2606
```

The yield to maturity of the T-bill can be calculated using the `yldtbill` function:

```
Yield = yldtbill(Settle, Maturity, Face, Price)
```

The yield of the T-bill in this example is:

```
Yield = 0.0386
```

The bond equivalent yield of the T-bill is

```
BEYield = beytbill(Settle, Maturity, Discount)
```

where `Discount` is the discount rate of the T-bill, which can be computed from the `discrate` function:

```
Discount = discrate(Settle, Maturity, Face, Price, Basis)
```

In this example, Basis = 2 (Actual/360 day-count convention) so that

```
Discount = 0.0379
```

as expected (as initially given) so that the bond equivalent yield is as follows:

```
BEYield = 0.0392
```

1.4 HEDGING DEBT POSITIONS

Hedging a Future 91-Day T-Bill Investment with T-Bill Call

Following Johnson (2004), suppose a treasurer expects higher short-term rates in June but is still concerned about the possibility of lower rates. To be able to gain from the higher rates and yet still hedge against lower rates, the treasurer could buy a June call option on a spot T-bill or a June option on a T-bill futures. For example, suppose there was a June T-bill futures option with an exercise price of 90 (strike price $X = 975,000$), price of 1.25 ($C = \$3,125$), and June expiration (on both underlying futures and option) occurring at the same time a $5,000,000 cash inflow is to be received. To hedge the 91-day investment with this call, the treasurer would need to buy 5.128205 calls (assume divisibility) at a cost of $16,025.64:

$$n_c = \frac{CF_T}{X} = \frac{\$5,000,000}{\$975,000} = 5.128205 \text{ contracts}$$
$$Cost = (5.128205)(\$3,125) = \$16,025.64$$
$$\pi_c = 5.1282085[Max(S_T - \$975,000, 0) - \$3,125]$$

If T-bill rates were lower at the June expiration, then the treasurer would profit from the calls that she could use to defray part of the cost of the higher priced T-bills. As shown in Table 1.4, if the spot discount rate on T-bills is 10% or less, the treasurer could buy 5.112 91-day spot T-bills with the $5 million cash inflow and profit from the calls, locking in a YTM of 9.3% on the $5 million investment. On the other hand, if T-bill rates were higher, then the treasurer would benefit from lower spot prices, while the losses on the call would be limited to just the $16,025.64 cost of the calls. In this case, for spot discount rates above 10%, the treasurer could buy more T-bills the higher the rates, resulting in higher yields as rates increase. Thus, for the cost of the call options, the treasurer can lock in a minimum YTM on the $5 million June investment of 9.3%, with the chance to earn a higher rate if short-term rates increase.

Table 1.4 Hedging \$5M *CF* in June with June T-Bill Futures Call

Call: X = 90 (975,000), C = 1.25 (\$3,125), n = 5.1282051				
1	2	3	4	5
Spot Rate: R	Spot Price	Profit/Loss	nTB	YTM
8	980000	9615.38456	5.112	0.093
8.5	978750	3205.12819	5.112	0.093
9	977500	-3205.1282	5.112	0.093
9.5	976250	-9615.3846	5.112	0.093
9.75	975625	-12820.513	5.112	0.093
10	975000	-16025.641	5.112	0.093
10.25	974375	-16025.641	5.115	0.096
10.5	973750	-16025.641	5.118	0.098
10.75	973125	-16025.641	5.122	0.101
11	972500	-16025.641	5.125	0.104
11.25	971875	-16025.641	5.128	0.107

Source: Johnson, S. (2004)

Note that if the treasurer wanted to hedge a 182-day investment instead of 91-days with calls, then similar to the futures hedge, she would need to buy both June and September T-bill futures calls. At the June expiration, the manager would then close both positions and invest the \$5,000,000 inflow plus (minus) the call profits (losses) in 182-day spot T-bills.

Short Hedge: Managing the Maturity Gap

Short hedges are used when corporations, municipal governments, financial institutions, dealers, and underwriters are planning to sell bonds or borrow funds at some future date and want to lock in the rate. The converse of the preceding example would be a money market manager who, instead of buying T-bills, was planning to sell her holdings of T-bills in June when the current bills would have maturities of 91 days or 182 days. To lock in a given revenue, the manager would go short in June T-bill futures (if she plans to sell 91-day bills) or June and September futures (if she planned to sell 182-day bills). If short-term rates increase (decrease), causing T-bill prices to decrease (increase), the money manager would receive less (more) revenue from selling the bills, but would gain (lose) when she closed the T-bill futures contracts by going long in the expiring June (and September) contract.

Another important use of short hedges is in locking in the rates on future debt positions. As an example, consider the case of a small bank with a maturity gap problem in which its short-term loan portfolio has an average maturity greater than the maturity of the CDs that it is using to finance the loans. Specifically, suppose in June, the bank makes loans of \$1 million, all with maturities of 180 days. To finance the loan, though, suppose the bank's customers prefer 90-day CDs to 180-day, and as a result, the bank has to sell \$1 million worth of 90-day CDs at a rate equal to the current LIBOR of 8.258%. Ninety days later (in September) the bank would owe $\$1,019,758 = (\$1,000,000)(1.08258)^{90/365}$; to finance this debt, the bank would have to sell \$1,019,758 worth of 90-day CDs at the LIBOR at

that time. In the absence of a hedge, the bank would be subject to market risk. If short-term rates increase, the bank would have to pay higher interest on its planned September CD sale, lowering the interest spread it earns (the rate from $1 million 180-day loans minus interest paid on CDs to finance them); if rates decrease, the bank would increase its spread.

Suppose the bank is fearful of higher rates in September and decides to minimize its exposure to market risk by hedging its $1,019,758 CD sale in September with a September Eurodollar futures contract trading at IMM = 92.1. To hedge the liability, the bank would need to go short in 1.03951 September Eurodollar futures (assume perfect divisibility):

$$f_0^{Sept} = \frac{100 - (7.9)(0.25)}{100}(\$1,000,000) = \$981,000$$

$$n_f = \frac{\$1,019,758}{\$981,000} = 1.03951 \text{ Short Eurodollar Contracts}$$

At a futures price of $981,000, the bank would be able to lock in a rate on its September CDs of 8.1%.

$$YTM_f^{Sept} = \left(\frac{\$1,000,000}{\$981,000}\right)^{365/91} - 1 = 0.081$$

$$YTM_{182} = \left[(1.0825)^{90/365}(1.081)^{90/36}\right]^{365/180} - 1 = 0.0817.$$

With this rate and the 8.25% rate it pays on its first CDs, the bank would pays 8.17% on its CDs over the 180-day period: That is, when the first CDs mature in September, the bank will issue new 90-day CDs at the prevailing LIBOR to finance the $1,019,758 first CD debt plus (minus) any loss (profit) from closing its September Eurodollar futures position. If the LIBOR in September has increased, the bank will have to pay a greater interest on the new CDs, but it will realize a profit from its futures that, in turn, will lower the amount of funds it needs to finance at the higher rate. On the other hand, if the LIBOR is lower, the bank will have lower interest payments on its new CDs, but it will also incur a loss on its futures position and therefore will have more funds that need to be financed at the lower rates. The impact that rates have on the amount of funds needed to be financed and the rate paid on them will exactly offset each other, leaving the bank with a fixed debt amount when the September CDs mature in December. As Table 1.5 shows, where the bank's December

Table 1.5

Sept LIBOR (R)	**0.075**	**0.085**
(2) $S_T^{CD} = f_T^{Sept} = \$1M/(1+R)^{90/365}$	$982,236	$979,640
(3) $\pi_t = [981,000 - f_T]1.0391$	-$1,378	$1,413
(4) Debt on June CD	$1,019,758	$1,019,758
(5) Total Funds to finance for next 90 days: Row (4) − Row (3)	$1,021,136	$1,018,345
(6) Debt at end of next 90 days: Row (5) $(1+R)^{90/365}$	$1,039,509	$1,039,509
(7) Rate for 180-day period: $R_{CD}^{180} = [\$1,039,509/\$1,000,000]^{365/180} - 1$	8.17%	8.17%

Source: Johnson, R.S.

liability (the liability at the end of the initial 180-day period) is shown to be $1,039,509 given September LIBOR rate scenarios of 7.5% and 8.7% (this will be true at any rate).

Note that the debt at the end of 180 days of $1,039,509 equates to a September 90-day rate of 8.1% and a 180-day rate for the period of 8.17%:

$$YTM_f^{Sept} = \left[\frac{\$1,039,509}{\$1,019,758}\right]^{365/90} - 1 = 0.081$$

$$R_{CD180} = \left[\frac{\$1,039,509}{\$1,000,000}\right]^{365/180} - 1 = 0.0817$$

Maturity Gap and the Carrying Cost Model

In the preceding example, we assumed the bank's maturity gap was created as a result of the bank's borrowers wanting 180-day loans and its investors/depositors wanting 90-day CDs. Suppose, though, that the bank does not have a maturity gap problem; that is, it can easily sell 180-day CDs to finance its 180-day loan assets and 90-day CDs to finance its 90-day loans. However, suppose that the September Eurodollar futures price was above its carrying cost value. In this case, the bank would find that instead of financing with a 180-day spot CD, it would be cheaper if it financed its 180-day June loans with synthetic 180-day CDs formed by selling 90-day June CDs rolled over three months later with 90-day September CDs, with the September CD rate locked in with a short position in the September Eurodollar futures contract. For example, if the June spot 180-day CDs were trading 96 to yield 8.63%, then the carrying cost value on the September Eurodollar contract would be 97.897 ($97.897 = 96(1.08258)^{90/365}$). If the September futures price were 98.1, then the bank would find it cheaper to finance the 180-day loans with synthetic 180-days CDs with an implied futures rate of 8.17% than with 180-day spot CDs at a rate of 8.63%. On the other hand, if the futures price is less than the carrying cost value, then the rate on the synthetic 180-day CDs would exceed the spot 180-day CD rate, and the bank would obtain a lower financing rate with the spot CDs. Finally, if the carrying cost model governing Eurodollar futures prices holds, then the rate of the synthetic will be equal to rate on the spot; in this case, the bank would be indifferent to its choice of financing.

Managing the Maturity Gap with Eurodollar Put

Instead of hedging its future CD sale with Eurodollar futures, the bank could alternatively buy put options on either a Eurodollar or T-bill or put options on a Eurodollar futures or T-bill futures. In the preceding case, suppose the bank decides to hedge its September CD sale by buying a September T-bill futures put with an expiration coinciding with the maturity of its June CD, an exercise price of 90 ($X = \$975,000$), and a premium of.5 ($C = \$1,250$). With the September debt from the June CD of $1,019,758, the bank would need to buy 1.046 September T-bill futures puts at a total cost of $1,307 to hedge the rate it pays on its September CD:

$$n_p = \frac{CF_T}{X} = \frac{\$1,019,758}{\$975,000} = 1.0459056 \text{ puts}$$

$$Cost = (1.046)(\$1,250) = \$1,307$$

If rates at the September expiration are higher such that the discount rate on T-bills is greater than 10%, then the bank will profit from the puts. This profit would serve to reduce part of the $1,019,750 funds it would need to finance the maturing June CD that, in turn, would help to negate the higher rate it would have to pay on its September CD. As shown in Table 1.6, if the T-bill discount yield is 10% or higher and the bank's 90-day CD rate is 0.25% more than the yield on T-bills, then the bank would be able to lock in a debt obligation 90 days later of $1,047,500, for an effective 180-day rate of 9.876%.

On the other hand, if rates decrease such that the discount rate on a spot T-bill is less than 10%, then the bank would be able to finance its $1,047,500 debt at lower rates, while its losses on its T-bill futures puts would be limited to the premium of $1,307. As a result, for lower rates, the bank would realize a lower debt obligation 90 days later and therefore a lower rate paid over the 180-day period. Thus, for the cost of the puts, hedging the maturity gap with puts allows the bank to lock in a maximum rate paid on debt obligations with the possibility of paying lower rates if interest rates decrease.

Table 1.6

Maturity Gap Hedged with T-Bill Puts

(1) R_D	(2) S_Y	(3) $Rate_{CD}$	(4) π_P	(5) *Debt on CD*	(6) *Funds Needed* (5) - (4)	(7) *Debt 90 Days Later* $(6)[1+(3)]^{90/345}$	(8) *Rate* $[(7)/1M]^{365/180}$
7%	$982,500	.07588	–1307	1,019,750	1,021,065	1,039,646	8.203%
8%	$980,000	.08690	–1307	1,019,750	1,021,065	1,042,262	8.756%
9%	$977,500	.09807	–1307	1,019,750	1,021,065	1,044,893	9.313%
10%	$975,000	.10940	–1307	1,019,750	1,021,065	1,047,500	9.867%
11%	$972,500	.12080	1307	1,019,750	1,018,451	1,047,500	9.867%
12%	$970,000	.13240	3922	1,019,750	1,015,836	1,047,500	9.867%

Assume 90-day CD rate is .25% greater than T-bill rate:

$$Rate_{CD} = \left[\frac{\$1M}{S_T}\right]^{365/91} + .0025 - 1$$

$$\pi_P = 1.4059056[Max[\$975{,}000 - S_T, 0] - \$1307$$

Source: Johnson, R.S. (2004)

Short Hedge: Hedging a Variable-Rate Loan

As a second example of a short naive hedge, consider the case of a corporation obtaining a one-year, $1 million variable-rate loan from a bank. In the loan agreement, suppose the loan starts on date 9/20 at a rate of 9.5% and then is reset on 12/20, 3/20, and 6/20 to equal the spot LIBOR (annual) plus 150 basis points (.015 or 1.5%) divided by four: $(\text{LIBOR} + .015)/4$.

To the bank, this loan represents a variable-rate asset, which it can hedge against interest rate changes by issuing 90-day CDs each quarter that are tied to the LIBOR. To the

corporation, though, the loan subjects them to interest rate risk (unless they are using the loan to finance a variable-rate asset). To hedge this variable-rate loan, though, the corporation could go short in a series of Eurodollar futures contracts—Eurodollar strip. For this case, suppose the company goes short in contracts expiring at 12/20, 3/20, and 6/20 and trading at the prices shown in Table 4.9.

Table 1.7

T	12/20	3/20	6/20
IMM Index	91.5	91.75	92
f_0(per \$100 Par)	97.875	97.9375	98

The locked-in rates obtained using Eurodollar futures contracts are equal to 100 minus the IMM index plus the basis points on the loans:

$$\begin{aligned}
&\textit{Locked-in Rate} = [100 - \textit{IMM}] + [\textit{BP}/100] \\
&12/20 : R_{12/20} = [100 - 91.5] + 1.5\% = 10\% \\
&\quad 3/20 : R_{3/20} = [100 - 91.75] + 1.5\% = 9.75\% \\
&\quad 6/20 : R_{6/20} = [100 - 92] + 1.5\% = 9.5\%
\end{aligned}$$

For example, suppose on date 12/20, the assumed spot LIBOR is 9%, yielding a settlement IMM index price of 91 and a closing futures price of 97.75 per \$100 face value. At that rate, the corporation would realize a profit of \$1,250 from having it short position on the 12/20 futures contract. That is:

$$f_0 = \frac{(100 - (100 - 91.5))(0.25)}{100}(\$1,000,000) = \$978,750$$

$$f_T = \frac{(100 - (100 - 91))(0.25)}{100}(\$1,000,000) = \$977,500$$

$$\text{Profit on 12/20 contract} = \$978,750 - \$977,500 = \$1,250$$

At the 12/20 date, though, the new interest that the corporation would have to pay for the next quarter would be set at \$26,250:

$$\begin{aligned}
&12/20 \text{ Interest} = [(\text{LIBOR} + .015)/4](\$1,000,000) \\
&12/20 \text{ Interest} = [(.09 + .015)/4](\$1,000,000) \\
&12/20 \text{ Interest} = \$26,250
\end{aligned}$$

Subtracting the futures profit from the \$26,250 interest payment (and ignoring the time value factor), the corporation's hedged interest payment for the next quarter is \$25,000. On an annualized basis, this equates to a 10% interest on a \$1 million loan, the same rate as the locked-in rate:

$$\text{Hedged Rate} = R^A = \frac{4(\$25,000)}{\$1,000,000} = 0.10$$

On the other hand, if the 12/20 LIBOR were 8%, then the quarterly interest payment would be only $\$23,750((.08+.015)/4)(\$1,000,000) = \$23,750$). This gain to the corporation, though, would be offset by a \$1,250 loss on the futures contract (i.e., at 8%, $f_T = \$980,000$, therefore, profit on the 12/20 contract is $\$978,750 - \$980,000 = -\$1,250$). As a result, the total quarterly debt of the company again would be $\$25,000(\$23,750 + \$1,250)$. Ignoring the time value factor, the annualized hedged rate the company pays would again be 10%. Thus, the corporation's short position in the 12/20 Eurodollar futures contract at 91.5 enables it to lock-in a quarterly debt obligation of \$25,000 and 10% annualized borrowing rate. If the LIBOR is at 9% on date 12/20, the company will have to pay \$26,250 on its loan the next quarter, but it will also have a profit on its 12/20 Eurodollar futures of \$1,250, which it can use to defray part of the interest expenses, yielding an effective hedged rate of 10%. The interest payments, futures profits, and effective interests are summarized:

$$
\begin{aligned}
12/20 : \textit{LIBOR} &= 9\% \\
\text{Futures:} & \\
\text{Settlementprice: } S_T &= 100 - \textit{LIBOR} \\
S_T &= 100 - 9(.25) = 97.75 \\
\pi_f &= \frac{97.875 - 97.75}{100}(\$1M) = \$1,250 \\
\text{Interest} &= \frac{\textit{LIBOR} + 150\textit{BP}}{4}(\$1M) \\
&= \frac{.09 + .015}{4}(\$1M) = \$26,250 \\
\text{EffectiveInterest} &= \$26,250 - \$1,250 = \$25,000 \\
\text{EffectiveRate} &= R^A = \frac{4(\$25,000)}{\$1M} = .10
\end{aligned}
$$

If the LIBOR is at 6% on date 12/20, the company will have to pay only \$18,750 on its loan the next quarter, but it will also have to cover a loss on its 12/20 Eurodollar futures of \$6,250. The payment of interest and the loss on the futures yields an effective hedged rate of 10%:

$$
\begin{aligned}
12/20 : \textit{LIBOR} &= 6\% \\
\text{Futures:} & \\
\text{Settlementprice : } S_T &= 100 - \textit{LIBOR} \\
S_T &= 100 - 6(.25) = 98.5 \\
\pi_f &= \frac{97.875 - 98.5}{100}(\$1M) = -\$6,250 \\
\text{Interest} &= \frac{\textit{LIBOR} + 150\textit{BP}}{4}(\$1M) \\
&= \frac{.06 + .015}{4}(\$1M) = \$18,750
\end{aligned}
$$

$$\text{EffectiveInterest} = \$18,750 + \$6,250 = \$25,000$$
$$\text{EffectiveRate} = R^A = \frac{4(\$25,000)}{\$1M} = .10$$

Given the other locked-in rates, the one-year fixed rate for the corporation on its variable-rate loan hedged with the Eurodollar futures contracts would therefore be 9.6873%:

$$\textit{Loan Rate} = \left[(1.095)^{.25}(1.10)^{.25}(1.0975)^{.25}(1.095)^{.25}\right]^1 - 1 = .096873.$$

Note, in practice, the corporation could have obtained a one-year fixed-rate loan. With futures contracts, though, the company now has a choice of taking either a fixed-rate loan or a synthetic fixed-rate loan formed with a variable-rate loan and short position in a Eurodollar futures contract, whichever is cheaper. Also, note that the corporation could have used a series of Eurodollar puts or futures puts to hedge its variable-rate loan. With a put hedge, each quarter the company would be able to lock in a maximum rate on its loan with the possibility of a lower rate if interest rates decrease.

1.5 Bond and Swap Duration, Modified Duration, and DV01

Duration (also known as Macauley's duration) is the present-value-weighted average time (in years) to maturity of the cash flow payments of a fixed income security. If C_i is the total cash flow payment at time t_i, then duration is computed as

$$\textit{Duration} = \frac{\sum_{i=1}^{n} \frac{t_i C_i}{(1+y_i)^{t_i}}}{B} \tag{1.4}$$

where n = number of cash flows, t = time to cash flow (in years), y_i is the simple-compound discount yield at time t_i, and $B = \sum_{i=1}^{n} \frac{C_i}{(1+y_i)^{t_i}}$ is the bond price.[12] At maturity, the cash flow includes both the principal face value and coupon payment. Yield modified duration[13] divides the duration number by $(1+1/\textit{YTM})$, where *YTM* is the yield to maturity of the bond. However, the *YTM* used to calculate the yield modified duration of a swap is the par swap rate (e.g., the swap rate that makes the present value of the swap zero).

To calculate the duration (modified duration) of a swap, we compute the modified duration of the long leg minus the modified duration of the short leg. Duration for the fixed leg is the present-value-weighted average maturity of the cash flows, whereas duration of the floating leg is the time to the next reset for the floating leg. The duration calculations (for both sides) divided by $1/(1+\textit{YTM})$ is the modified duration for each side. The modified duration of an interest rate swap leg can be calculated like a bond

$$\textit{MD} = \frac{\textit{Leg DV01}}{\textit{Leg PV}} * 10,000 \tag{1.5}$$

where DV01 (sometimes denoted PV01 in the literature) is the dollar value of a basis point change of the leg[14] and PV is the present value of the leg. This calculation should always be

positive. An alternative, but equivalent, calculation to compute modified duration for each leg is to compute

$$MD = \frac{Leg\ DV01}{(Notional\ Value + Market\ Value)} * 10,000 \tag{1.6}$$

where the Market Value is the leg PV. This alternative computation is appropriate if the leg PV does not include the final exchange of principal. However, the leg DV01 must include the notional exchange to give a bond-like duration, so the leg DV01 must also include notional.

The difference between the modified durations of both legs, the net PV, is then taken to compute the modified duration of the swap:

$$\text{Swap MD} = \text{MD of Receive Leg} - \text{MD of Pay Leg} \tag{1.7}$$

Note that if the market value (leg PV) of the swap is zero, then duration is computed. Thus, for a par swap where the market value is zero, the duration (referred to as *deal risk* in Bloomberg) is equal to the modified duration.

For forward-starting swaps—e.g., a 10-year swap that starts in 7 years—typically a discounted swap modified duration is computed by adjusting equation (1.7) by a factor that is very close to the discount factor for the start date of the swap. An excellent approximation for this discount factor is provided by the NPV of the floating leg (including the final notional exchange), divided by the notional of the trade:

$$\text{Discounted Swap MD} = \text{Swap MD} \times \text{Floating Leg PV}/\text{Notional} \tag{1.8}$$

Note that quantitative differences arise between the yield modified duration (which assumes a flat discount rate) and the DV01 formulation in (1.5) or (1.6), based on various factors such as a sloping yield curve, instrument valuations away from par, instruments with long tenors, and forward-starting instruments. With an upward-sloping discount curve, the PV of the largest cash flow (at maturity) affects duration the most. Because the yield is lower than the discount rate at maturity, the value of the yield MD is higher than the value of the DV01 MD. The effect of the upward-sloping curve is amplified with a longer tenor, because the largest cash flow at maturity is even more different when discounted using the curve for DV01 MD and using the yield for yield MD; thus, the yield MD is greater than the DV01 MD. A downward-sloping discount curve has the opposite affect because the yield is higher than the discount rate at maturity.

Large NPVs during a tenor with an upward-sloping yield curve result from a coupon much different from the discount rate. The present value of the coupon cash flows have a larger weight in the bond price (PV). For DV01 MD, the PV of coupons in the long end decreases more when the curve is shifted by one basis point, compared to yield MD where coupons at the end are discounted at a lower rate (the yield) and are less sensitive to interest rate changes. Also, the DV01 MD is calculated by dividing by a higher value of bond PV so that the yield MD is greater than the DV01 MD. For a forward starting swap with an upward-sloping yield curve, yield MD is calculated as the sum of the yield MDs of two bonds with opposite signs—a long position in a bond starting today and maturing in T_2 years, and a short position in bond starting today and maturity in T_1 years, where

$0 < T_1 < T_2$. The cash flows affecting yield are those between $T_1 + 1$ and T_2 years, which are discounted at a yield lower than the discount rates at the long end of the curve. As a result, yield MD is higher than the DV01 MD.

Hedging Bond Portfolios

Consider the following bond portfolio example shown in Table 1.8.[15]

Of the Treasury bonds in the portfolio, the 5% of August 2011 was the current on-the-run 10-year, while the 6.5% of February 2010 became the cheapest-to-deliver (CTD) for the CBOT 10-year Treasury note futures when the data was recorded. The 5.625% of May 08 had recently been CTD. The coupons of the 11 corporate bonds in Table 1.8 range from a high of 9.375% to a low of 6.15% and have maturities from August 2007 to August 2011. With the exception of American Standard, all of these issues are investment grade credits where the credit ratings range from the S&P AA- of the Transamerica issue to the BBB- of the News America Holdings and Litton Industries issue. American Standard is a BB+ credit.

Table 1.8 shows weighted average durations for the two sectors and for the portfolio as a whole. The Treasury sector shows that this $262.34 million holding has a duration of 6.56 years. The $436.25 million corporate sector holding has a duration of 5.39 years, and the $698.59 million portfolio has a duration of 5.83 years.

Fixed income securities with different coupons and maturities respond differently to yield changes. This price sensitivity to yield change can be captured in terms of the dollar value of a basis point (DV01) for a given security. To find the DV01 for an individual fixed income security, given a modified duration and a full price, solve the following:

Table 1.8

Issuer	Coupon	Maturity	YTM	Modified Duration (years)	DV01 ($)	Par Amount ($ millions)	Full Price ($000s)	S&P Credit Rating
Treasury	6.5	2/15/10	4.55	6.56	0.0748	67	76,387	
Treasury	5.625	5/15/08	4.64	5.48	0.0588	92	98,762	
Treasury	5	8/15/11	4.57	7.77	0.0807	84	87,192	
Treasury Sector				6.56			262,341	
Time Warner Enterprises	8.18	8/15/07	5.47	4.72	0.0540	82	93,710	BBB+
Texas Utilities	6.375	1/1/08	6.19	5.06	0.0517	30	30,678	BBB
Rockwell International	6.15	1/15/08	6.14	5.13	0.0521	52	52,837	A
Transamerica Corporation	9.375	3/1/08	6.34	4.93	0.0573	15	17,445	AA-
Coastal Corporation	6.5	6/1/08	6.76	5.25	0.0528	30	30,153	BBB
United Airlines	6.831	9/1/08	5.99	5.51	0.0579	38	39,949	A-
Burlington Northern Santa Fe	7.34	9/24/08	5.67	5.36	0.0606	3	3,390	A+
News America Holdings	7.375	10/17/08	6.56	5.34	0.0575	30	32,292	BBB-
Litton Industries	8	10/15/09	5.70	5.81	0.0647	63	66,834	BBB-
American Standard Inc.	7.625	2/15/10	7.59	6.10	0.0615	41	41,332	BB+
Caterpillar Inc.	9.375	8/15/11	6.01	6.79	0.0853	22	27,628	A+
Corporate Sector				5.39			436,248	
Portfolio				5.83			698,589	

$$DV01 = \frac{(Duration/100) * Full\ \text{Price}}{100}$$

For instance, using this formula with the price and duration date in Table 1.8, we see that a $98,762,000 position in the Treasury 5.625% of May 08, with its 5.48 duration, has a DV01 of $54,122, while an $87,192,000 position in the Treasury 5% of August 11, with its 7.77 duration, has a $67,748 DV01. These DV01s predict that 1 basis point rise in yield would drive the value of the 5.625% of May 08 down $54,122, while the same yield change would drive the value of the smaller holding in the 5% of August 11 down $67,748. Clearly, the 5% of August 11 is more sensitive to yield changes than the 5.625% of May 08 is.

The portfolio is subject to inflation, interest rate, and credit risk that could sharply erode the value of the holdings and make it difficult to liquidate them at acceptable prices. The structuring of hedge positions then requires that the futures position be ratioed to the positions, and then requires that the futures position be ratioed to the position in the security being hedged in order that the two positions will respond equally to a given yield change. Given that the DV01 of the 10-year Treasury note futures contract was $72.50 on the day these dates were recorded (the 10-year swap futures has a $77.00 DV01), we can solve for the optimal hedge ratio to see that it would take 747 contracts to hedge the 5.625% of May 08 and 934 contracts to hedge the 5% of August 11:

$$\frac{5.625\%\ \text{of May 08 DV01}}{\text{futures DV01}} = \frac{54,122}{72.50} = 747\ \text{contracts}$$

$$\frac{5\%\ \text{of August 11 DV01}}{\text{futures DV01}} = \frac{67,748}{72.50} = 934\ \text{contracts}$$

To structure a portfolio hedge, we can use a weighted average duration for the Treasury sector, the corporate sector, or the entire holding, and then use the total full price of the relevant sectors to find DV01s. Table 1.9 shows that, based on the data in Table 1.8, the Treasury sector has a $172,096 DV01, the corporate segment has a $235,138 DV01, and

Table 1.9

		DV01 ($)
Treasury Sector		172,096
Corporate Sector		235,138
Portfolio		402,277
10-year Treasury Futures		72.50
10-year Swap Futures		77.00
Hedge Ratios		
	To Hedge	*Hedge Ratio*
1	Full portfolio with 10-yr. T-note futures	5,618
2	Full portfolio with swap futures	5,289
3	Treasury sector with 10-yr. T-note futures	2,374
4	Corporate sector with swap futures	3,054

the entire portfolio has a $407,277 DV01 (note that the portfolio DV01 does not equal the sum of the two sector DV01s due to rounding error).

Plugging in these DV01s into the hedge ratio formulas yields the optimal hedge ratios. Hedging the entire portfolio with Treasury futures requires a short position in 5,618 contracts:

$$\frac{\text{Portfolio DV01 } 407,277}{\text{10-year Treasury Futures DV01 } 72.50} = 5,618 \text{ contracts}$$

Similarly, hedging the entire portfolio with swap futures requires a short position of 5,289 10-year swap futures contracts:

$$\frac{\text{Portfolio DV01 } 407,277}{\text{10-year Swap Futures DV01 } 77.00} = 5,289 \text{ contracts}$$

Finally, an optimal hedging strategy where the Treasury sector is hedged with Treasury futures and the corporate sector is hedged with swap futures requires a short position in 2,375 10-year Treasury futures contracts and a short position in 3,054 10-year swap futures contracts:

$$\frac{\text{Treasury Sector DV01 } 172,096}{\text{10-year Treasury Futures DV01 } 72.50} = 2,375 \text{ contracts}$$

$$\frac{\text{Corporate Sector DV01 } 235,138}{\text{10-year Swap Futures DV01 } 72.50} = 3,054 \text{ contracts}$$

It is important to note that these hedges are static in nature—they only apply at a given point in time based on current market data. Because Treasury, corporate yields, and swap rate cause shifts in the DV01s, these hedges should be monitored constantly and rebalanced based on the changes in interest rates and thus amount of risk exposure.

Hedging and portfolio performance can be measured by scenario and prediction analysis of yield shifts. During the summer of 2001, the 10-year Treasury-swap rate credit spread increased 40 basis points. Corporate yields generally followed the direction of the swap rate. Should that occur again where the swap rate and corporate yields rise 60 bps while Treasury yields rise only 20 bps, the underlying portfolio will incur an approximate $17.55 million loss. Table 1.10 shows that the sector DV01s predict that $3,441,920 of the loss will come from the Treasury sector and $14,108,280 of it will come from the corporate sector. The Treasury futures hedge (according to the prediction of $72.50 DV01) will respond to the 20 bp change in the Treasury yield and generate an $8,146,000 gain (see Table 1.10, Scenario 1). Based on this scenario, a hedge mismatch loss of $9,404,100 is incurred as a result of the fact that the Treasury futures hedge can be expected to offset less than half the portfolio loss.

Alternatively, swap futures can be used to hedge the entire portfolio. This strategy has the advantage of responding to the larger change in the swap rate and corporate yields. However, as Table 1.10 Scenario 2 shows, the relevant DV01s predict that this hedge will generate a futures gain much larger than the loss the underlying portfolio will incur. As Table 1.10, Scenario 3 shows, the relevant DV01s predict that these two hedge positions will generate a total futures gain of $17,551,780. Thus, Scenario 3 appears far more promising

Table 1.10

Underlying Portfolio Result

Sector	DV01 ($)	Yield Change (bps)	Number of Contracts	Gain/Loss ($)
Treasury	172,096	20		-3,441,920
Corporate	235,138	60		-14,108,280
Portfolio				-17,550,200

Scenario 1 - Hedge Entire Portfolio with Treasury Futures

Sector	DV01 ($)	Yield Change (bps)	Number of Contracts	Gain/Loss ($)
10-year T-note	72.50	20	5,618	8,146,100
Portfolio				-17,550,200
Hedge Mistatch				-9,404,100

Scenario 2 - Hedge Entire Portfolio with Swap Futures

Sector	DV01 ($)	Yield Change (bps)	Number of Contracts	Gain/Loss ($)
10-year swap	77.00	60	5,289	24,435,180
Portfolio				-17,550,200
Hedge Mistatch				6,884,980

Scenario 3 - Hedge Treasury Sector with Treasury Futures, Hedge Corporate Sector with Swap Futures

Sector	DV01 ($)	Yield Change (bps)	Number of Contracts	Gain/Loss ($)
10-year T-note	72.50	20	2,374	3,442,300
10-year swap	77.00	60	3,054	14,109,480
Total Hedge Result				17,550,780
Portfolio				-17,550,200
Portfolio Mismatch				1,380
Treasury to 10-year T-note Mismatch				380
Corporate to Swap Mismatch				1,200

than either of the first two hedges given the minimal hedge mismatch of $1,580 (which represents less than a basis point and is far less than normal bid-ask spreads, and so for practical purposes represents a good offset). It highlights the importance of using exact or like (highly correlated) sector hedges to hedge underlying sectors: hedge the Treasury sector with 10-year Treasury note futures and the corporate sector with 10-year swap futures.

The large variation in Scenario 1 and Scenario 2, compared with the small hedge mismatch in Scenario 3, can be explained by the impact of basis risk that cannot be captured by single sector hedges of an entire underlying fixed income portfolio with multiple sectors. A 10-year Treasury futures is not as correlated with the corporate sector as the 10-year swap futures. In other words, swaps correlate more closely with corporates than Treasuries do. For example, consider a corporate-swap and corporate-Treasury regression using the TransAmerica Corp. 9.375% March 2008 corporate bond in Table 1.8. The corporate-swap correlation regression coefficient R^2 is 0.9134, while the corporate-Treasury R^2 is 0.7966, as shown in Figure 1.1.

The 0.9134 R^2 of the corporate-swap regression suggests that the variability of the swap rate accounts for 91.34% of the variability in the corporate yield. In contrast, the 0.7966

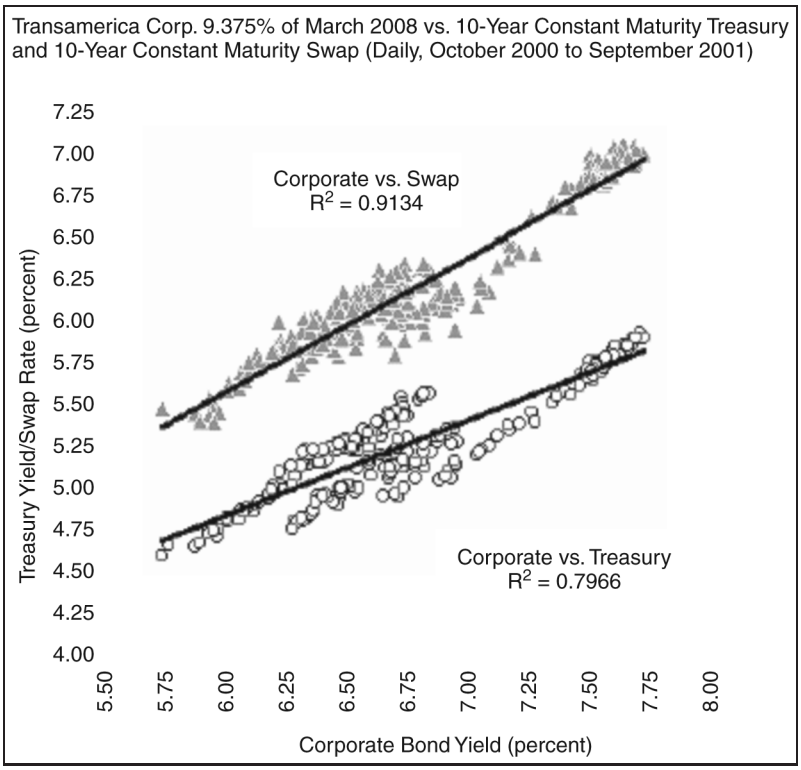


Figure 1.1 *Reprinted by permission of the Board of Trade of the City of Chicago, Inc., © 2001. All Rights Reserved.*

R^2 of the corporate-Treasury regression suggests that the variability of the Treasury yield accounts for slightly less than 80% of the variability of the yield of the corporate bond.

1.6 Term Structure of Rates

Term structure modeling is essential for valuation of fixed income securities and derivatives as a means for quantifying the relationship between either price or yield among a set of securities that differ only in the timing of their cash flows or their term until maturity. Term structure models describe the evolution of interest rates over time. The relationship expressed by the term structure is traditionally the par-coupon yield relationship, though in general, the term structure could be the discount function, the spot-yield curve, or some other price-yield relationship.

The set of securities that define a term structure is known as the *reference set*, which may be a set of U.S. Treasuries, agency debentures, off-the-run Treasury issues, interest rate swaps, or single-A rated corporate bonds, for instance. The n-year zero coupon yield, also known as the n-year spot rate, is the interest rate on an investment that is earned for a period of time starting today and lasting for n years. There is widespread usage of the par yield

curve for the Treasury market so that many market sectors are defined from a reference set derived from the Treasury market;[16] for example, "the reference set that defines the agency debenture market is a set of yield spreads on the on-the-run Treasuries, so that the five-year debenture issued by an agency may be priced at par to yield 15 basis points more than the current five-year Treasury issue."[17] Other important yield curves include the forward rate curve and swap curve. Forward interest rates (future spot rates) are the rates of interest implied by current spot rates for periods of time in the future. The swap curve shows the fixed rate that is to be paid for receiving a floating rate such as three-month Libor, for a given swap maturity.

There are typically three types of term structures. The (coupon) yield curve is the yield-to-maturity structure of coupon bonds. The zero-coupon yield or spot-rate curve is the term structure of discount rates of zero-coupon bonds. The forward rate curve is the term structure of forward rates implicit in zero-coupon discount rates.

1.7 BOOTSTRAP METHOD

To construct the zero-coupon yield curve when spot rates cannot be observed directly, use the *bootstrap method* to extract it from observable coupon-bearing bond (cash) prices, swap rates, and interest rate futures. The bootstrap method, an iterative numerical method for extracting spot rates from previously computed spot rates and observable bond prices, can be used to construct the discount rate curve. Denote y_n as the yield to maturity of an n-period bond of maturity and d_n as the spot (discount) rate of the n-period maturity bond. The method is based on the idea that a coupon bond can be decomposed into the sum of zero-coupon bonds:

$$\begin{aligned} P_n &= \frac{c/m}{1+y_n/2} + \frac{c/m}{(1+y_n/2)^2} + \ldots + \frac{F+c/m}{(1+y_n/2)^n} \\ &= \frac{c/m}{1+d_n/2} + \frac{c/m}{(1+d_n/2)^2} + \ldots + \frac{F+c/m}{(1+d_n/2)^n} \\ &= Z_1 + Z_2 + \ldots + Z_n \end{aligned}$$

The standard procedure for bootstrapping the Treasury yield curve, known as *recursive stripping*, is as follows:

1. Obtain the current price of U.S. Treasuries (and/or Eurodollar futures).
2. Solve for the yield on the one-period zero-coupon bond, z_1.
3. Solve for z_2, given the price of the two-period bond and z_1 calculated in step 2.
4. Solve for z_3, given the price of the three-period bond and z_1 and z_2.
5. Continue until all the spot rates have been calculated.

Bootstrapping is a process whereby one starts with known data points and then solves for unknown data points using an underlying arbitrage theory. Every coupon bond can be valued as a package of zero-coupon bonds that mimic its cash flow and risk characteristics.

By mapping yields-to-maturity for each theoretical zero-coupon bond, to the dates spanning the investment horizon, one can create a theoretical zero-rate curve.

The zero spot rates are typically quoted on a bond equivalent basis.

One can use the bootstrap method to extract the zero coupon curve, which can, in turn, be used to construct a discount rate curve to discount cash flows. The generation of these curves typically starts with a series of on-the-run and selected off-the-run issues as input. All cash flows are used to construct the spot curve, and rates between maturities (for these coupons) are linearly interpolated.

For each bond maturity, we solve for the current yield that equates the current bond maturity with the sum of its discounted cash flows based on the determination of the yields of all the shorter maturity bonds. For instance, to determine the short rate yield on a bond that matures in n-years, we need to solve for d_n, which can be achieved only if all the yields for bonds maturing at periods $i = 1, ..., n-1$ are determined. The method is called the bootstrap method because it determines yields (and thus the short rate curve) by building on top or "bootstrapping" from all previously determined yields:

$$P_n = \sum_{i=1}^{n-1} \frac{C_n}{(1+d_i)^i} + \frac{C_n + FV}{(1+d_n/2)^n}$$

For a zero-coupon bond, one can compute the (continuously computed) discount rate d_c from a discount spot rate with compounding n times per annum[18] via the following equation:

$$d_c = n \ln\left(1 + \frac{c_n}{n}\right)$$

Consider the Treasury bills and bonds listed in Table 1.11. We assume bond prices that mature every six months are available out to five years.

When computing the spot rates, on-the-run treasuries are typically used as they are more liquid and normally trade close to par, thereby mitigating any tax-biases associated with discount or premium bonds. Unfortunately, on-the-run securities are available only at 0.5-, 1-, 2-, 3-, 5-, and 10-year maturities. Of the methods used to obtain spot rates between

Table 1.11

Bond Price	Annual Coupon	Semiannual Period	Maturity (Years)	Period Coupon
102.2969	6.125	1	0.5	3.0625
104.0469	6.25	2	1.0	3.125
104.0000	5.25	3	1.5	2.625
103.5469	4.75	4	2.0	2.375
109.5156	7.25	5	2.5	3.625
111.1719	7.5	6	3.0	3.750
122.4844	10.75	7	3.5	5.375
119.6094	9.375	8	4.0	4.687
111.3281	7.0	9	4.5	3.500
108.7031	6.25	10	5.0	3.125

these values, exponential cubic splines[19] is the most common. However, recursive stripping is also common. We assume the maturity face value is 100.

To bootstrap the curve, we start by computing the discount spot rate that equates the cash flows to the 0.5-year maturity bond price of 102.27:

$$102.27 = \frac{(3.0625 + 100)}{(1 + d_1/2)}$$

or solving for d_1:

$$d_1 = (103.0625/102.27 - 1) * 2 = 0.014968$$

We compute the discount sum for semiannual period 1:

$$\textit{DiscountSum}1 = 1/(1 + d_1) = 1/(1.04968) = 0.9853$$

We next solve for the one-year spot rate that gives the bond price at semiannual period 2:

$$\begin{aligned} 104.05 &= \frac{3.125}{(1 + d_1)} + \frac{103.125}{(1 + d_2/2)^2} \\ &= 3.125 * \textit{DiscountSum}1 + 103.125/(1 + d_2/2)^2 \\ &= 2 * \left(\left(\frac{\textit{PeriodCoupon}2 + \textit{FaceValue}}{\textit{BondPrice}2 - \textit{PeriodCoupon}2 * \textit{DiscountSum}1} \right)^{1/2} - 1 \right) \end{aligned}$$

so that

$$d_2 = 2.1250\%.$$

We compute the discount sum for semiannual period 2:

$$\begin{aligned} \textit{DiscountSum}2 &= \textit{DiscountSum}1 + 1/(1 + d_2)^2 \\ &= 0.9853 + 1/(1.02125)^2 \\ &= 1.94407 \end{aligned}$$

We next solve for the sport rate that equates the cash flows to the 1.5-year maturity bond price (third semiannual period):

$$104 = 2.625 * \textit{DiscountSum}2 + \frac{102.625}{(1 + d_3/2)^3}$$

or solving

$$\begin{aligned} d_3 &= 2 * \left(\left(\frac{\textit{PeriodCoupon}3 + \textit{FaceValue}}{\textit{BondPrice}3 - \textit{PeriodCoupon}3 * \text{DiscountSum}2} \right)^{1/3} - 1 \right) \\ &= 2.4822\%. \end{aligned}$$

So that, in general, the bootstrapped spot rate for semiannual period n is given by

$$d_n = 2 * \left(\left(\frac{\textit{PeriodCoupon}(n) + \textit{FaceValue}}{\textit{BondPrice}(n) - \textit{PeriodCoupon}(n) * \textit{DiscountSum}(n-1)} \right)^{1/n} - 1 \right)$$

where

$$\textit{DiscountSum}(n-1) = \textit{DiscountSum}(n-2) + 1/(1+d_2)^{n-1}.$$

If we continue the process, we find the discount sums and spot rates shown in Table 1.12, which yield the term structure of rates shown in Figure 1.2.

In practice, bootstrapping requires the input of T-bills, Treasury bond, Treasury note futures, Eurodollar futures, or coupon-bearing bond prices. One must interpolate the spot rate yields for specific maturities not available from the available market prices. For instance, some bond tables list net yields for bonds in a sequence of one, three, and five years. Interpolation would be used to determine the yield for the second and fourth year. In effect, interpolation is a process of trial and error utilizing a numerical method like the Newton-Raphson method. What complicates the procedure is that day count conventions need to be taken into account for computing accrued interest on coupon-bearing bonds.

Table 1.12

Period	Discount Sum	Spot Rate
1	0.985252547	1.4968%
2	1.944068996	2.1250%
3	2.87315116	2.4822%
4	3.76649457	2.8597%
5	4.623301565	3.1391%
6	5.442647715	3.3766%
7	6.227073953	3.5295%
8	6.975043088	3.6966%
9	7.682592791	3.9187%
10	8.359670683	3.9767%

1.8 BOOTSTRAPPING IN MATLAB

There are three ways to import data into Matlab: (1) loading the data as a "flat file" in ASCII format and then converting it to a numerical matrix; (2) loading the data first into Excel and then using ExcelLink to pass numeric matrices to the Matlab workspace; or (3) using the Matlab Database Toolbox to pull the data from an ODBC-compliant database.

Matlab, via the Financial Toolbox, provides two bootstrapping functions: `zbtprice`, which bootstraps the zero curve from coupon-bond data-given prices and `zbtyield`, which bootstraps the zero curve from coupon-bond data-given yields. Suppose we want to bootstrap the yield curve in Matlab using the corporate bond data in Figure 1.1 to get the zero rates at the maturity rates of the bonds. Assume the settlement date is January 2, 2006.

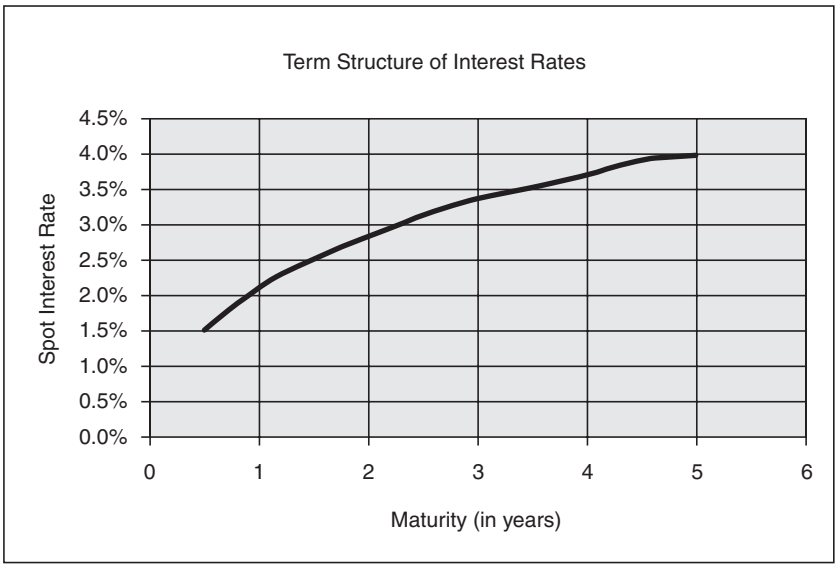


Figure 1.2 Term structure of rates

```
% Bonds = [Maturity CouponRate FaceValue]
Bonds = [datenum('15-Aug-2007')  0.08180   100;
         datenum('01-Jan-2008')  0.06375   100;
         datenum('15-Jan-2008')  0.06150   100;
         datenum('01-Mar-2008')  0.09375   100;
         datenum('01-Jun-2008')  0.06500   100;
         datenum('01-Sep-2008')  0.06831   100;
         datenum('24-Sep-2008')  0.07340   100;
         datenum('17-Oct-2008')  0.07375   100;
         datenum('15-Oct-2009')  0.08000   100;
         datenum('15-Feb-2010')  0.07625   100;
         datenum('15-Aug-2011')  0.09375   100];

Yields = [0.0547; 0.0619; 0.0604; 0.0634; 0.0676; 0.0599; 0.0567;
          0.0656; 0.0670; 0.0750; 0.0601];

Prices = [114.28; 102.26; 101.61; 116.30; 100.51; 105.13; 113.00;
          107.64; 111.39; 100.81; 125.58];

Settle = datenum('02-Jan-2006');

[ZeroRates, CurveDates] = zbtyield(Bonds,Yields,Settle)

ZeroRates =

    0.0547
    0.0623
```

```
    0.0607
    0.0641
    0.0683
    0.0600
    0.0565
    0.0661
    0.0676
    0.0769
    0.0586

datestr(CurveDates) =

15-Aug-2007
01-Jan-2008
15-Jan-2008
01-Mar-2008
01-Jun-2008
01-Sep-2008
24-Sep-2008
17-Oct-2008
15-Oct-2009
15-Feb-2010
15-Aug-2011
```

We could also bootstrap using the bond prices:

```
      [ZeroRates] = zbtprices(Bonds,Prices,Settle);

ZeroRates =

    0.0251
    0.0581
    0.0582
    0.0400
    0.0665
    0.0554
    0.0362
    0.0533
    0.0548
    0.0810
    0.0469
```

It should be noted that these zero rates are not risk-free discount rates because the coupon bonds are not risk free. To compute risk-free zero-rates, T-bills, Treasury notes, and Treasury bonds should be used.

1.9 BOOTSTRAPPING IN EXCEL

Consider the Excel spreadsheet[20] (ZC.xls) shown in Figure 1.3 with a worksheet called "Bootstrap," which takes zero coupon rates as input and interpolates between maturity dates. To view the Visual Basic code, click on Tools > Macro > Visual Basic Editor.

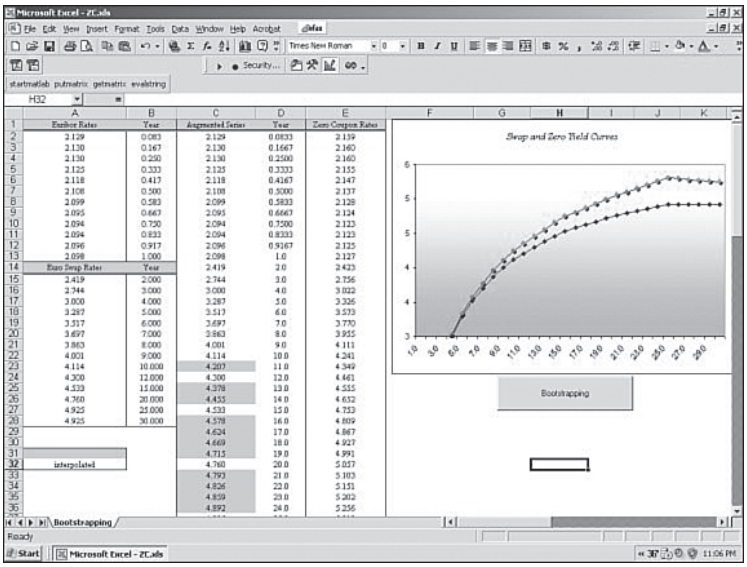


Figure 1.3 Interpolating zero coupon rates between maturity dates

```
Public ZC(1 To 100) As Double                // stores zero coupon rates
Public swaprate(1 To 100) As Double          // stores swap rates
Public dataswap(1 To 100) As Double
```

VB code in ZC.xls: Sub_interpolation_swap

```
Sub interpolation_swap()

  ReadRates_over10y  'Load swap rates over 10 years

  For i = 1 To 12
    Sheets("Bootstrapping").Cells(i + 1, 3) =
      Sheets("Bootstrapping").Cells(i + 1, 1)
    Sheets("Bootstrapping").Cells(i + 1, 4) =
      Sheets("Bootstrapping").Cells(i + 1, 2)
  Next i

  For i = 1 To 9
    Sheets("Bootstrapping").Cells(i + 13, 3) =
      Sheets("Bootstrapping").Cells(i + 14, 1)
    Sheets("Bootstrapping").Cells(i + 13, 4) =
      Sheets("Bootstrapping").Cells(i + 14, 2)

  Next i

  Sheets("Bootstrapping").Cells(24, 3) = swaprate(12)
  Sheets("Bootstrapping").Cells(27, 3) = swaprate(15)
  Sheets("Bootstrapping").Cells(32, 3) = swaprate(20)
  Sheets("Bootstrapping").Cells(37, 3) = swaprate(25)
```

```
Sheets("Bootstrapping").Cells(42, 3) = swaprate(30)

'Start the interpolation procedure for each knot

swaprate(11) = (swaprate(10) + swaprate(12)) * 0.5
Sheets("Bootstrapping").Cells(23, 3) = swaprate(11)


For i = 13 To 14
  swaprate(i) = swaprate(12) + ((swaprate(15) - swaprate(12)) *
    ((i - dataswap(2)) / (dataswap(3)
- dataswap(2))))
  Sheets("Bootstrapping").Cells(i + 12, 3) = swaprate(i)
Next i


For i = 16 To 19
swaprate(i) = swaprate(15) + ((swaprate(20) - swaprate(15)) *
  ((i - dataswap(3)) / (dataswap(4) - dataswap(3))))
  Sheets("Bootstrapping").Cells(i + 12, 3) = swaprate(i)
Next i


For i = 21 To 24
  swaprate(i) = swaprate(20) + ((swaprate(25) - swaprate(20)) *
    ((i - dataswap(4)) / (dataswap(5)
- dataswap(4))))
  Sheets("Bootstrapping").Cells(i + 12, 3) = swaprate(i)
Next i


For i = 26 To 29
swaprate(i) = swaprate(25) + ((swaprate(30) - swaprate(25)) *
  ((i - dataswap(5)) / (dataswap(6)
- dataswap(5))))
  Sheets("Bootstrapping").Cells(i + 12, 3) = swaprate(i)
Next i

End Sub

Sub ReadRates_over10y()

Dim i As Integer
Dim j As Integer
Sheets("Bootstrapping").Select

For i = 1 To 6
  dataswap(i) = Sheets("Bootstrapping").Cells(i + 22, 2)
  swaprate(dataswap(i)) = Sheets("Bootstrapping").Cells(i + 22, 1)
Next i

End Sub

Sub ZC_Rates()

Rates_Load
```

```
  ' Uniform the day-count convention between spot and swap rates
    (both 30/360)

  For i = 1 To 12

    ZC(i) = Sheets("Bootstrapping").Cells(i + 1, 3) * 365 / 360

  Next i

  For j = 13 To 41

  dummy_sum = 0

    For i = 1 To j - 12
          dummy_sum = dummy_sum + (swaprate(j) / 100) /
            ((1 + (ZC(11 + i) / 100)) ^ dataswap(11 + i))
    Next i

    ZC(j) = ((((1 + (swaprate(j) / 100)) / (1 - dummy_sum)) ^
      (1 / (dataswap(11 + i)))) - 1) * 100

  Next j

  For i = 1 To 41
    Sheets("Bootstrapping").Cells(i + 1, 5) = ZC(i)
  Next i

End Sub

Sub Rates_Load()

  Dim i As Integer

  For i = 1 To 41
    dataswap(i) = Sheets("Bootstrapping").Cells(i + 1, 4)
              swaprate(i) = Sheets("Bootstrapping").Cells(i + 1, 3)
  Next i

End Sub
```

1.10 General Swap Pricing in Matlab

The Matlab Fixed-Income Toolbox[21] contains functions that perform swap pricing and portfolio hedging. The Fixed-Income Toolbox contains the function **`liborfloat2fixed`**, which computes a fixed-rate par yield that equates the floating-rate side of a swap to the fixed-rate side. The solver sets the present value of the fixed side to the present value of the floating side without having to line up and compare fixed and floating periods.

The following assumptions are used for floating-rate input:

- LIBOR rates are quarterly—for example, that of Eurodollar futures.
- Effective date is the first third Wednesday after the settlement date.
- All delivery dates are spaced three months apart.

- All periods start on the third Wednesday of delivery months.
- All periods end on the same dates of delivery months, three months after the start dates.
- Accrual basis of floating rates is actual/360.
- Applicable forward rates are estimated by interpolation in months when forward-rate data in not available.

The following assumptions are used for floating-rate output:

- Design allows you to create a bond of any coupon, basis, or frequency, based upon the floating-rate input.
- The start date is a valuation date—that is, a date when an agreement to enter into a contract by the settlement date is made.
- Settlement can be on or after the start date. If it is after, a forward fixed-rate contract results.
- Effective date is assumed to be the first third Wednesday after settlement—the same date as that of the floating rate.
- The end date of the bond is a designated number of years away, on the same day and month as the effective date.
- Coupon payments occur on anniversary dates. The frequency is determined by the period of the bond.
- Fixed rates are not interpolated. A fixed-rate bond of the same present value as that of the floating-rate payments is created.

To compute par fixed-rate of swap given three-month LIBOR data, the following function is used:

```
[FixedSpec, ForwardDates, ForwardRates] =
       liborfloat2fixed(ThreeMonthRates, Settle, Tenor, StartDate,
       Interpolation, ConvexAdj, RateParam, InArrears, Sigma,
       FixedCompound, FixedBasis)
```

The input arguments are shown in Table 1.13.
The output is as follows:

`FixedBasis` computes forward rates, dates, and the swap fixed rate.
`FixedSpec` specifies the structure of the fixed-rate side of the swap:

- `Coupon`: Par-swap rate.
- `Settle`: Start date.
- `Maturity`: End date.

Table 1.13

`ThreeMonthRates`	Three-month Eurodollar futures data or forward rate agreement data. (A forward rate agreement stipulates that a certain interest rate applies to a certain principal amount for a given future time period.) An n-by-3 matrix in the form of `[month year IMMQuote]`. The floating rate is assumed to compound quarterly and to accrue on an actual/360 basis.
`Settle`	Settlement date of swap. Scalar.
`Tenor`	Life of the swap. Scalar.
`StartDate`	(Optional) Scalar value to denote reference date for valuation of (forward) swap. This in effect allows forward swap valuation. Default = `Settle`.
`Interpolation`	(Optional) Interpolation method to determine applicable forward rate for months when no Eurodollar data is available. Default is `'linear'` or 1. Other possible values are `'Nearest'` or 0, and `'Cubic'` or 2.
`ConvexAdj`	(Optional) Default = 0 (off). 1 = on. Denotes whether futures/-forward convexity adjustment is required. Pertains to forward rate adjustments when those rates are taken from Eurodollar futures data.
`RateParam`	(Optional) Short-rate model's parameters (Hull-White) `[a S]`, where the short-rate process is: $dr = [\theta(t) - ar]dt + Sdz$ Default = `[0.05 0.015]`.
`InArrears`	(Optional) Default = 0 (off). Set to 1 for on. If on, the routine does an automatic convexity adjustment to forward rates.
`Sigma`	(Optional) Overall annual volatility of caplets.
`FixedCompound`	(Optional) Scalar value. Compounding or frequency of payment on the fixed side. Also, the reset frequency. Default = 4 (quarterly). Other values are 1, 2, and 12.
`FixedBasis`	(Optional). Scalar value. Basis of the fixed side. 0 = actual/actual, 1 = 30/360 (SIA, default), 2 = actual/360, 3 = actual/365, 4 = 30/360 (PSA), 5 = 30/360 (ISDA), 6 = 30/360 (European), 7 = act/365 (Japanese).

- `Period`: Frequency of payment.
- `Basis`: Accrual basis.

`ForwardDates` are dates corresponding to `ForwardRates` (all third Wednesdays of the month, spread three months apart). The first element is the third Wednesday immediately after `Settle`.

`ForwardRates` are forward rates corresponding to the forward dates, quarterly compounded, and on the actual/360 basis. To preserve input integrity, tenor is rounded upward to the closest integer. Currently traded tenors are 2, 5, and 10 years. The function assumes

that floating-rate observations occur quarterly on the third Wednesday of a delivery month. The first delivery month is the month of the first third Wednesday after `Settle`. Floating-side payments occur on the third-month anniversaries of observation dates.

Example 3

Use the supplied `EDdata.xls` file as input to a `liborfloat2fixed` computation.

```
[EDFutData, textdata] = xlsread('EDdata.xls');
Settle                = datenum('15-Oct-2002');
Tenor                 = 2;

[FixedSpec, ForwardDates, ForwardRates] = ...
liborfloat2fixed(EDFutData, Settle, Tenor)

FixedSpec =

      Coupon: 0.0222
      Settle: '16-Oct-2002'
    Maturity: '16-Oct-2004'
      Period: 4
       Basis: 1

ForwardDates =

      731505  (16-Oct-2002)
      731596  (15-Jan-2003)
      731687  (16-Apr-2003)
      731778  (16-Jul-2003)
      731869  (15-Oct-2003)
      731967  (21-Jan-2004)
      732058  (21-Apr-2004)
      732149  (21-Jul-2004)

ForwardRates =

    0.0177
    0.0166
    0.0170
    0.0188
    0.0214
    0.0248
    0.0279
    0.0305
```

Table 1.14 shows Eurodollar data on Friday 11, 2002 that we will used for bootstrapping the yield curve and for computing swap rates in MATLAB.

Using this data, you can compute 1-, 2-, 3-, 4-, 5-, 7-, and 10-year swap rates with the toolbox function `liborfloat2fixed`. The function requires you to input only Eurodollar data, the settlement date, and tenor of the swap. Matlab then performs the required computations.

To illustrate how this function works, first load the data contained in the supplied Excel worksheet, `EDdata.xls`.

Table 1.14

Eurodollar Data on Friday 11, 2002					
Month	**Year**	**Settle**	**Month**	**Year**	**Settle**
10	2002	98.21	6	2007	94.88
11	2002	98.26	9	2007	94.74
12	2002	98.3	12	2007	94.595
1	2003	98.3	3	2008	94.48
2	2003	98.31	6	2008	94.375
3	2003	98.275	9	2008	94.28
6	2003	98.12	12	2008	94.185
9	2003	97.87	3	2009	94.1
12	2003	97.575	6	2009	94.005
3	2004	97.26	9	2009	93.925
6	2004	96.98	12	2009	93.865
9	2004	96.745	3	2010	93.82
12	2004	96.515	6	2010	93.755
3	2005	96.33	9	2010	93.7
6	2005	96.135	12	2010	93.645
9	2005	95.955	3	2011	93.61
12	2005	95.78	6	2011	93.56
3	2006	95.63	9	2011	93.515
6	2006	95.465	12	2011	93.47
9	2006	95.315	3	2012	93.445
12	2006	95.16	6	2012	93.41
3	2007	95.025	9	2012	93.39

Source: Matlab

```
[EDRawData, textdata] = xlsread('EDdata.xls');
```

Extract the month from the first column and the year from the second column. The rate used as proxy is the arithmetic average of rates on opening and closing.

```
Month = EDRawData(:,1);
Year = EDRawData(:,2);
IMMData = (EDRawData(:,3);
EDFutData = [Month, Year, IMMData];
```

Next, input the current date.

```
Settle = datenum('11-Oct-2002');
```

To compute for the two-year swap rate, set the tenor to 2.

```
Tenor = 2;
```

Finally, compute the swap rate with `liborfloat2fixed`.

```
[FixedSpec, ForwardDates, ForwardRates] =...
     liborfloat2fixed(EDFutData, Settle, Tenor)
```

Matlab returns a par-swap rate of 2.23% using the default setting (quarterly compounding and 30/360 accrual), and forward dates and rates data (quarterly compounded), comparable to 2.17% of Friday's average broker data in Table H15 of *Federal Reserve Statistical Release* (http://www.federalreserve.gov/releases/h15/update/).

```
FixedSpec =

       Coupon: 0.0223
       Settle: '16-Oct-2002'
     Maturity: '16-Oct-2004'
        Period: 4
         Basis: 1

datestr(ForwardDates) =

      731505  (16-Oct-2002)
      731596  (15-Jan-2003)
      731687  (16-Apr-2003)
      731778  (16-Jul-2003)
      731869  (15-Oct-2003)
      731967  (21-Jan-2004)
      732058  (21-Apr-2004)
      732149  (21-Jul-2004)

ForwardRates =

    0.0179
    0.0170
    0.0177
    0.0196
    0.0222
    0.0255
    0.0285
    0.0311
```

In the `FixedSpec` output, note that the swap rate actually goes forward from the third Wednesday of October 2002 (October 16, 2002), five days after the original `Settle` input (October 11, 2002). This, however, is still the best proxy for the swap rate on `Settle`, as the assumption merely starts the swap's effective period and does not affect its valuation method or its length.

The correction suggested by Hull and White improves the result by turning on convexity adjustment as part of the input to `liborfloat2fixed`. (See Hull, J., *Options, Futures, and Other Derivatives*, 4th Edition, Prentice Hall, 2000.) For a long swap—e.g., five years or more—this correction could prove to be substantial.

The adjustment requires additional parameters:

- `StartDate`, which you make the same as `Settle` (the default) by providing an empty matrix `[ ]` as input.

- `ConvexAdj` to tell `liborfloat2fixed` to perform the adjustment.

- `RateParam`, which provides the parameters `a` and `S` as input to the Hull-White short-rate process.

- Optional parameters `InArrears` and `Sigma`, for which you can use empty matrices `[]` to accept the Matlab defaults.

- `FixedCompound`, with which you can facilitate comparison with values cited in Table H15 of *Federal Reserve Statistical Release* by turning the default quarterly compounding into semiannual compounding, with the (default) basis of 30/360.

```
StartDate = [];
Interpolation = [];
ConvexAdj = 1;
RateParam = [0.03; 0.017];
FixedCompound = 2;
[FixedSpec, ForwardDates, ForwardRates] =...
liborfloat2fixed(EDFutData, Settle, Tenor, StartDate, Interpolation,
ConvexAdj, RateParam, [], [], FixedCompound)
```

This returns 2.21% as the two-year swap rate, quite close to the reported swap rate for that date. Analogously, Table 1.15 summarizes the solutions for 1-, 3-, 5-, 7-, and 10-year swap rates (convexity-adjusted and unadjusted).

Table 1.15

Calculated and Market Average Data of Swap Rates on Friday, October 11, 2002

Swap Length (Years)	Unadjusted	Adjusted	Table H15	Adjusted Error (Basis Points)
1	1.80%	1.79%	1.80%	-1
2	2.24%	2.21%	2.22%	-1
3	2.70%	2.66%	2.66%	0
4	3.12%	3.03%	3.04%	-1
5	3.50%	3.37%	3.36%	+1
7	4.16%	3.92%	3.89%	+3
10	4.87%	4.42%	4.39%	+3

Source: Matlab

To compute the duration of a LIBOR-based interest rate swap, we use the **`liborduration`** function:

```
[PayFixDuration GetFixDuration] = liborduration(SwapFixRate, Tenor,
Settle)
```

The input arguments are shown in Table 4.12.

Table 1.16

SwapFixRate	Scalar or column vector of swap fixed rates in decimal.
Tenor	Scalar or column vector indicating life of the swap in years. Fractional numbers are rounded upward.
Settle	Scalar or column vector of settlement dates.

The output arguments are as follows:

- PayFixDuration is the modified duration, in years, realized when entering pay-fix side of the swap.
- GetFixDuration is the modified duration, in years, realized when entering receive-fix side of the swap.

Example 4

Given the following data

```
SwapFixRate = 0.0383;
Tenor = 7;
Settle = datenum('11-Oct-2002');
```

compute the swap durations.

```
[PayFixDuration GetFixDuration] = liborduration(SwapFixRate,...
    Tenor, Settle)

PayFixDuration =

   -4.7567

GetFixDuration =

    4.7567
```

A swap can be valued in Matlab using a Black-Derman-Toy (BDT) tree or an HJM tree. The syntax is as follows:

```
[Price, PriceTree, CFTree, SwapRate] = swapbybdt(BDTTree, LegRate,
   Settle, Maturity, LegReset, Basis, Principal, LegType, Options)
```

The input arguments are shown Table 1.17.
The outputs are as follows:

- Price is the number of instruments (NINST)-by-1 expected prices of the swap at time 0.
- PriceTree is the tree structure with a vector of the swap values at each node.

- `CFTree` is the tree structure with a vector of the swap cash flows at each node.
- `SwapRate` is a `NINST`-by-1 vector of rates applicable to the fixed leg such that the swaps' values are zero at time 0. This rate is used in calculating the swaps' prices when the rate specified for the fixed leg in `LegRate` is `NaN`. `SwapRate` is padded with `NaN` for those instruments in which `CouponRate` is not set to `NaN`.

Table 1.17

`BDTTree`	Interest rate tree structure created by `bdttree`.
`LegRate`	Number of instruments (`NINST`)-by-2 matrix, with each row defined as: `[CouponRate Spread]` or `[Spread CouponRate]` `CouponRate` is the decimal annual rate. `Spread` is the number of basis points over the reference rate. The first column represents the receiving leg, while the second column represents the paying leg.
`Settle`	Settlement date. `NINST`-by-1 vector of serial date numbers or date strings. `Settle` must be earlier than or equal to `Maturity`.
`Maturity`	Maturity date. `NINST`-by-1 vector of dates representing the maturity date for each swap.
`LegReset`	(Optional) `NINST`-by-2 matrix representing the reset frequency per year for each swap. Default = `[1 1]`.
`Basis`	(Optional) `NINST`-by-1 vector representing the basis used when annualizing the input forward rate tree. Default = 0 (actual/actual).
`Principal`	(Optional) `NINST`-by-1 vector of the notional principal amounts. Default = `100`.
`LegType`	(Optional) `NINST`-by-2 matrix. Each row represents an instrument. Each column indicates if the corresponding leg is fixed (`1`) or floating (`0`). This matrix defines the interpretation of the values entered in `LegRate`. Default is `[1 0]`for each instrument.
`Options`	(Optional) Derivatives pricing options structure created with `derivset`.

Example 5

To price an interest rate swap with a fixed receiving leg and a floating paying leg, payments are made once a year, and the notional principal amount is $1,000,000. The values for the remaining parameters are as follows:

- Coupon rate for fixed leg: 0.15 (15%)
- Spread for floating leg: 10 basis points
- Swap settlement date: Jan. 01, 2000
- Swap maturity date: Jan. 01, 2003

Based on the preceding information, set the required parameters and build the `LegRate`, `LegType`, and `LegReset` matrices.

```
Settle = '01-Jan-2000';
Maturity = '01-Jan-2003';
Basis = 0;
Principal = 1000000;
LegRate = [0.15 10]; % [CouponRate Spread]
LegType = [1 0]; % [Fixed Float]
LegReset = [1 1]; % Payments once per year
```

We price the swap using the `BDTTree` included in the MAT-file `deriv.mat`. `BDTTree` contains the time and forward rate information needed to price the instrument.

```
load deriv;
```

Use `swapbybdt` to compute the price of the swap.

```
Price = swapbybdt(BDTTree, LegRate, Settle, Maturity,...
     LegReset, Basis, Principal, LegType)

Price = 73032
```

Example 6

Using the previous data, calculate the swap rate, the coupon rate for the fixed leg such that the swap price at time 0 is zero.

```
LegRate = [NaN 20];

[Price, PriceTree, CFTree, SwapRate] = swapbybdt(BDTTree,...
    LegRate, Settle, Maturity, LegReset, Basis, Principal, LegType)

Price =

 -2.8422e-014

PriceTree =

    FinObj: 'BDTPriceTree'
     tObs: [0 1 2 3 4]
     PTree: {1x5 cell}

CFTree =

    FinObj: 'BDTCFTree'
     tObs: [0 1 2 3 4]
     CFTree: {1x5 cell}

SwapRate =
             0.1210
```

A swap can also be valued using an HJM tree.

```
[Price, PriceTree, CFTree, SwapRate] = swapbyhjm(HJMTree, LegRate,
     Settle, Maturity, LegReset, Basis, Principal, LegType, Options)
```

Table 1.18

`HJMTree`	Forward rate tree structure created by `hjmtree`.
`LegRate`	Number of instruments (`NINST`)-by-2 matrix, with each row defined as: `[CouponRate Spread]` or `[Spread CouponRate]` `CouponRate` is the decimal annual rate. `Spread` is the number of basis points over the reference rate. The first column represents the receiving leg, while the second column represents the paying leg.
`Settle`	Settlement date. `NINST`-by-1 vector of serial date numbers or date strings. `Settle` must be earlier than or equal to `Maturity`.
`Maturity`	Maturity date. `NINST`-by-1 vector of dates representing the maturity date for each swap.
`LegReset`	(Optional) `NINST`-by-2 matrix representing the reset frequency per year for each swap. Default = `[1 1]`.
`Basis`	(Optional) `NINST`-by-1 vector representing the basis used when annualizing the input forward rate tree. Default = 0 (actual/actual).
`Principal`	(Optional) `NINST`-by-1 vector of the notional principal amounts. Default = `100`.
`LegType`	(Optional) `NINST`-by-2 matrix. Each row represents an instrument. Each column indicates if the corresponding leg is fixed (`1`) or floating (`0`). This matrix defines the interpretation of the values entered in `LegRate`. Default is `[1 0]` for each instrument.
`Options`	(Optional) Derivatives pricing options structure created with `derivset`.

The arguments of the `swapbyhjm` function are shown in Table 1.18.

The `Settle` date for every swap is set to the `ValuationDate` of the HJM tree. The swap argument `Settle` is ignored. This function also calculates the `SwapRate` (fixed rate) so that the value of the swap is initially zero. To do this, enter `CouponRate` as `NaN`.

Description

`[Price, PriceTree, CFTree, SwapRate] = swapbyhjm(HJMTree, LegRate, Settle, Maturity, LegReset, Basis, Principal, LegType)` computes the price of a swap instrument from an HJM interest rate tree.

`Price`	The number of instruments (`NINST`)-by-1 expected prices of the swap at time 0.
`PriceTree`	The tree structure with a vector of the swap values at each node.
`CFTree`	The tree structure with a vector of the swap cash flows at each node.
`SwapRate`	A `NINST`-by-1 vector of rates applicable to the fixed leg such that the swaps' values are zero at time 0. This rate is used in calculating the swaps' prices when the rate specified for the fixed leg in `LegRate` is `NaN`. `SwapRate` is padded with `NaN` for those instruments in which `CouponRate` is not set to `NaN`.

Example 7

Price an interest rate swap with a fixed receiving leg and a floating paying leg. Payments are made once a year, and the notional principal amount is $100. The values for the remaining parameters are as follows:

- Coupon rate for fixed leg: 0.06 (6%)
- Spread for floating leg: 20 basis points
- Swap settlement date: Jan. 01, 2000
- Swap maturity date: Jan. 01, 2003

Based on the preceding information, set the required parameters and build the `LegRate`, `LegType`, and `LegReset` matrices.

```
Settle = '01-Jan-2000';
Maturity = '01-Jan-2003';
Basis = 0;
Principal = 100;
LegRate = [0.06 20]; % [CouponRate Spread]
LegType = [1 0]; % [Fixed Float]
LegReset = [1 1]; % Payments once per year
```

Price the swap using the `HJMTree` included in the MAT-file `deriv.mat`. `HJMTree` contains the time and forward rate information needed to price the instrument.

```
load deriv;
```

Use `swapbyhjm` to compute the price of the swap.

```
[Price, PriceTree, CFTree] = swapbyhjm(HJMTree, LegRate,...
     Settle, Maturity, LegReset, Basis, Principal, LegType)

Price =

   3.6923

PriceTree =

    FinObj: 'HJMPriceTree'
      tObs: [0 1 2 3 4]
     PBush: {1x5 cell}

CFTree =

    FinObj: 'HJMCFTree'
      tObs: [0 1 2 3 4]
    CFBush: {[0] [1x1x2 double] [1x2x2 double] ... [1x8 double]}
```

1.11 Swap Pricing in Matlab Using Term Structure Analysis

This example illustrates some of the term structure analysis functions found in the Financial Toolbox. Specifically, it illustrates how to derive implied zero (*spot*) and forward curves from the observed market prices of coupon-bearing bonds. The zero and forward curves implied from the market data are then used to price an interest rate swap agreement. In an interest rate swap, two parties agree to a periodic exchange of cash flows. One of the cash flows is based on a fixed interest rate held constant throughout the life of the swap. The other cash flow stream is tied to some variable index rate. Pricing a swap at inception amounts to finding the fixed rate of the swap agreement. This fixed rate, appropriately scaled by the notional principal of the swap agreement, determines the periodic sequence of fixed cash flows. In general, interest rate swaps are priced from the forward curve such that the variable cash flows implied from the series of forward rates and the periodic sequence of fixed-rate cash flows have the same present value. Thus, interest rate swap pricing and term structure analysis are closely related.

Step 1. Specify values for the settlement date, maturity dates, coupon rates, and market prices for 10 U.S. Treasury bonds. This data allows us to price a five-year swap with net cash flow payments exchanged every six months. For simplicity, accept default values for the end-of-month payment rule (rule in effect) and day-count basis (actual/actual). To avoid issues of accrued interest, assume that all Treasury bonds pay semiannual coupons and that settlement occurs on a coupon payment date.

```
Settle = datenum('15-Jan-1999');

BondData = {'15-Jul-1999' 0.06000 99.93
            '15-Jan-2000' 0.06125 99.72
            '15-Jul-2000' 0.06375 99.70
            '15-Jan-2001' 0.06500 99.40
            '15-Jul-2001' 0.06875 99.73
            '15-Jan-2002' 0.07000 99.42
            '15-Jul-2002' 0.07250 99.32
            '15-Jan-2003' 0.07375 98.45
            '15-Jul-2003' 0.07500 97.71
            '15-Jan-2004' 0.08000 98.15};
```

`BondData` is an instance of a Matlab *cell array*, indicated by the curly braces {}.

Next, assign the date stored in the cell array to `Maturity`, `CouponRate`, and `Prices` vectors for further processing.

```
Maturity   = datenum(strvcat(BondData{:,1}));
CouponRate = [BondData{:,2}]';
Prices     = [BondData{:,3}]';
Period     = 2; % semiannual coupons
```

Step 2. Now that the data has been specified, use the term structure function **`zbtprice`** to bootstrap the zero curve implied from the prices of the coupon-bearing bonds. This implied zero curve represents the series of zero-coupon Treasury rates consistent with the prices of the coupon-bearing bonds such that arbitrage opportunities will not exist.

```
ZeroRates = zbtprice([Maturity CouponRate], Prices, Settle);
```

The zero curve, stored in `ZeroRates`, is quoted on a semiannual bond basis (the periodic, six-month, interest rate is simply doubled to annualize). The first element of `ZeroRates` is the annualized rate over the next six months, the second element is the annualized rate over the next 12 months, and so on.

Step 3. From the implied zero curve, find the corresponding series of implied forward rates using the term structure function **`zero2fwd`**.

```
ForwardRates = zero2fwd(ZeroRates, Maturity, Settle);
```

The forward curve, stored in `ForwardRates`, is also quoted on a semiannual bond basis. The first element of `ForwardRates` is the annualized rate applied to the interval between settlement and 6 months after settlement, the second element is the annualized rate applied to the interval from 6 months to 12 months after settlement, and so on. This implied forward curve is also consistent with the observed market prices such that arbitrage activities will be unprofitable. Because the first forward rate is also a zero rate, the first element of `ZeroRates` and `ForwardRates` are the same.

Step 4. Now that you have derived the zero curve, convert it to a sequence of discount factors with the term structure function **`zero2disc`**.

```
DiscountFactors = zero2disc(ZeroRates, Maturity, Settle);
```

Step 5. From the discount factors, compute the present value of the variable cash flows derived from the implied forward rates. For plain interest rate swaps, the notional principal remains constant for each payment date and cancels out of each side of the present value equation. The next line assumes unit notional principal.

```
PresentValue = sum((ForwardRates/Period) .* DiscountFactors);
```

Step 6. Compute the swap's price (the fixed rate) by equating the present value of the fixed cash flows with the present value of the cash flows derived from the implied forward rates. Again, because the notional principal cancels out of each side of the equation, it is simply assumed to be 1.

```
SwapFixedRate = Period * PresentValue / sum(DiscountFactors);
```

The output would be as follows:

```
 Zero Rates Forward Rates
   0.0614     0.0614
   0.0642     0.0670
   0.0660     0.0695
   0.0684     0.0758
   0.0702     0.0774
   0.0726     0.0846
   0.0754     0.0925
   0.0795     0.1077
```

```
   0.0827    0.1089
   0.0868    0.1239

Swap Price (Fixed Rate) = 0.0845
```

All rates are in decimal format. The swap price, 8.45%, would likely be the midpoint between a market-maker's bid/ask quotes.

Example 8

Consider a nine-year fixed for floating swap with a notional of $3.3 million and fixed coupon rate of 3.969%, a floating rate based on three-month T-bill, with a start date of March 28, 2004, effective date of March 29, 2004, and a maturity of March 28, 2013. Assume the fixed side is on a 30/360 day count while the floating leg is on an actual/360 day count with quarterly resets. Figure 1.4 displays the Bloomberg screen with the swap data terms.

Figures 1.5–1.7 show the swap cash flow payments and payment schedule. Figures 1.8 provides the swap curve. Figure 1.9, shows the risk measures, DV01, and duration, for both the pay and receive legs.

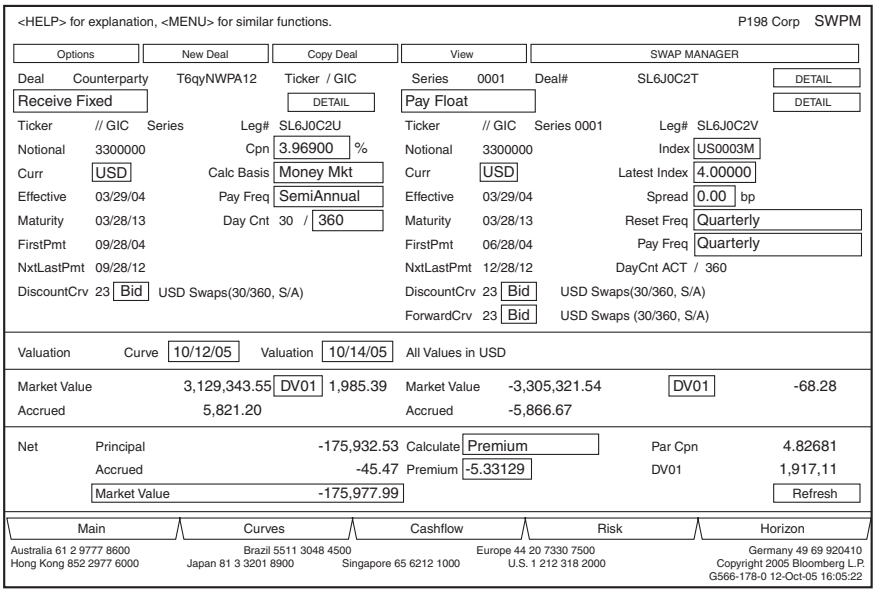
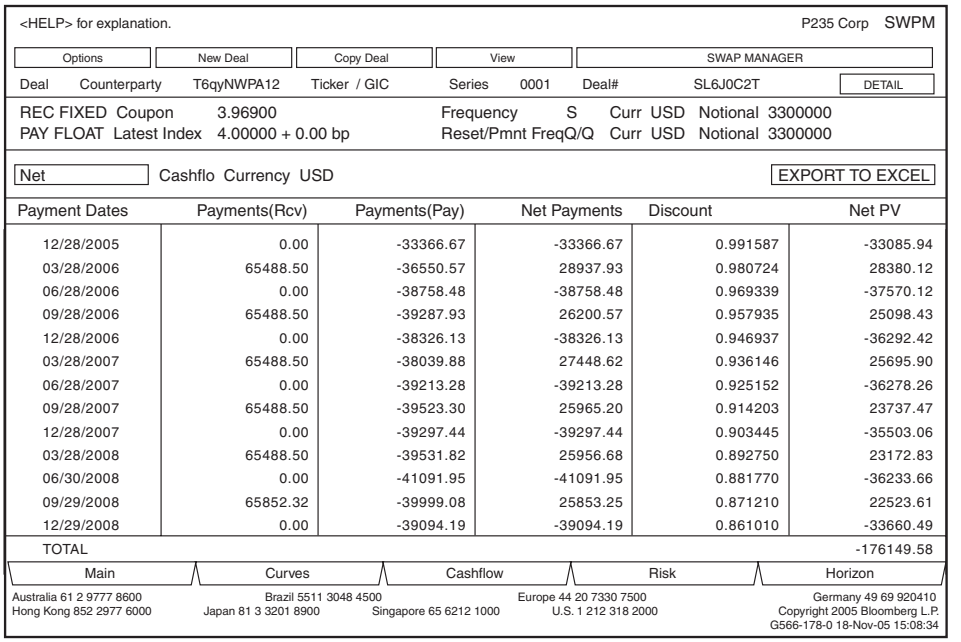


Figure 1.4 *Source: Reproduced with permission from Bloomberg.*



<HELP> for explanation. P235 Corp SWPM

Options | New Deal | Copy Deal | View | SWAP MANAGER

Deal Counterparty T6qyNWPA12 Ticker / GIC Series 0001 Deal# SL6J0C2T DETAIL

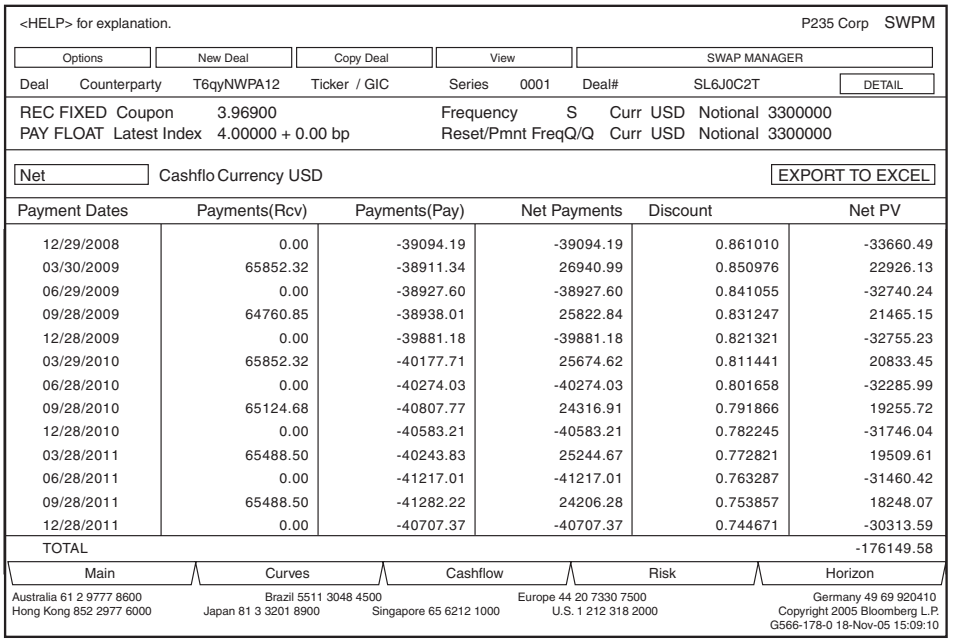
REC FIXED Coupon 3.96900 Frequency S Curr USD Notional 3300000
PAY FLOAT Latest Index 4.00000 + 0.00 bp Reset/Pmnt FreqQ/Q Curr USD Notional 3300000

Net Cashflo Currency USD EXPORT TO EXCEL

Payment Dates	Payments(Rcv)	Payments(Pay)	Net Payments	Discount	Net PV
12/28/2005	0.00	-33366.67	-33366.67	0.991587	-33085.94
03/28/2006	65488.50	-36550.57	28937.93	0.980724	28380.12
06/28/2006	0.00	-38758.48	-38758.48	0.969339	-37570.12
09/28/2006	65488.50	-39287.93	26200.57	0.957935	25098.43
12/28/2006	0.00	-38326.13	-38326.13	0.946937	-36292.42
03/28/2007	65488.50	-38039.88	27448.62	0.936146	25695.90
06/28/2007	0.00	-39213.28	-39213.28	0.925152	-36278.26
09/28/2007	65488.50	-39523.30	25965.20	0.914203	23737.47
12/28/2007	0.00	-39297.44	-39297.44	0.903445	-35503.06
03/28/2008	65488.50	-39531.82	25956.68	0.892750	23172.83
06/30/2008	0.00	-41091.95	-41091.95	0.881770	-36233.66
09/29/2008	65852.32	-39999.08	25853.25	0.871210	22523.61
12/29/2008	0.00	-39094.19	-39094.19	0.861010	-33660.49
TOTAL					-176149.58

Main | Curves | Cashflow | Risk | Horizon

Figure 1.5 *Source: Reproduced with permission from Bloomberg.*



<HELP> for explanation. P235 Corp SWPM

Options | New Deal | Copy Deal | View | SWAP MANAGER

Deal Counterparty T6qyNWPA12 Ticker / GIC Series 0001 Deal# SL6J0C2T DETAIL

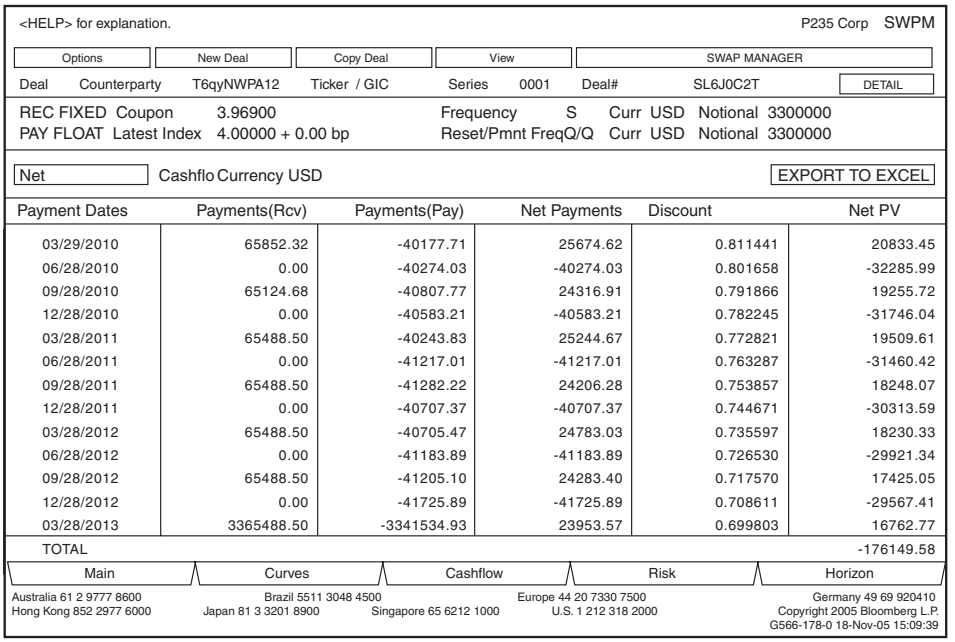
REC FIXED Coupon 3.96900 Frequency S Curr USD Notional 3300000
PAY FLOAT Latest Index 4.00000 + 0.00 bp Reset/Pmnt FreqQ/Q Curr USD Notional 3300000

Net Cashflo Currency USD EXPORT TO EXCEL

Payment Dates	Payments(Rcv)	Payments(Pay)	Net Payments	Discount	Net PV
12/29/2008	0.00	-39094.19	-39094.19	0.861010	-33660.49
03/30/2009	65852.32	-38911.34	26940.99	0.850976	22926.13
06/29/2009	0.00	-38927.60	-38927.60	0.841055	-32740.24
09/28/2009	64760.85	-38938.01	25822.84	0.831247	21465.15
12/28/2009	0.00	-39881.18	-39881.18	0.821321	-32755.23
03/29/2010	65852.32	-40177.71	25674.62	0.811441	20833.45
06/28/2010	0.00	-40274.03	-40274.03	0.801658	-32285.99
09/28/2010	65124.68	-40807.77	24316.91	0.791866	19255.72
12/28/2010	0.00	-40583.21	-40583.21	0.782245	-31746.04
03/28/2011	65488.50	-40243.83	25244.67	0.772821	19509.61
06/28/2011	0.00	-41217.01	-41217.01	0.763287	-31460.42
09/28/2011	65488.50	-41282.22	24206.28	0.753857	18248.07
12/28/2011	0.00	-40707.37	-40707.37	0.744671	-30313.59
TOTAL					-176149.58

Main | Curves | Cashflow | Risk | Horizon

Figure 1.6 *Source: Reproduced with permission from Bloomberg.*



<HELP> for explanation. P235 Corp SWPM

Options | New Deal | Copy Deal | View | SWAP MANAGER

Deal Counterparty T6qyNWPA12 Ticker / GIC Series 0001 Deal# SL6J0C2T DETAIL

REC FIXED Coupon 3.96900 Frequency S Curr USD Notional 3300000
PAY FLOAT Latest Index 4.00000 + 0.00 bp Reset/Pmnt FreqQ/Q Curr USD Notional 3300000

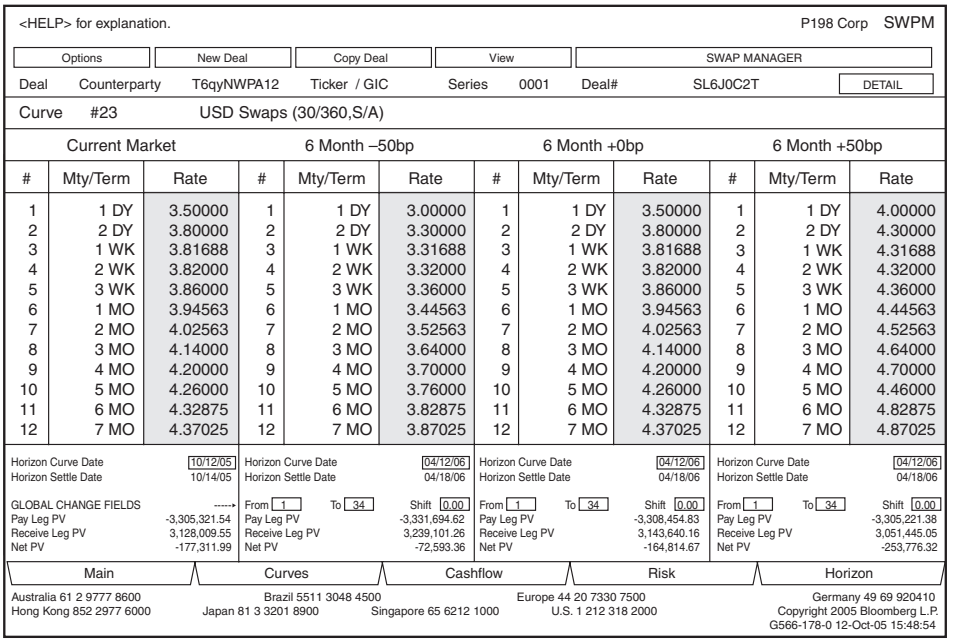
Net Cashflo Currency USD EXPORT TO EXCEL

Payment Dates	Payments(Rcv)	Payments(Pay)	Net Payments	Discount	Net PV
03/29/2010	65852.32	-40177.71	25674.62	0.811441	20833.45
06/28/2010	0.00	-40274.03	-40274.03	0.801658	-32285.99
09/28/2010	65124.68	-40807.77	24316.91	0.791866	19255.72
12/28/2010	0.00	-40583.21	-40583.21	0.782245	-31746.04
03/28/2011	65488.50	-40243.83	25244.67	0.772821	19509.61
06/28/2011	0.00	-41217.01	-41217.01	0.763287	-31460.42
09/28/2011	65488.50	-41282.22	24206.28	0.753857	18248.07
12/28/2011	0.00	-40707.37	-40707.37	0.744671	-30313.59
03/28/2012	65488.50	-40705.47	24783.03	0.735597	18230.33
06/28/2012	0.00	-41183.89	-41183.89	0.726530	-29921.34
09/28/2012	65488.50	-41205.10	24283.40	0.717570	17425.05
12/28/2012	0.00	-41725.89	-41725.89	0.708611	-29567.41
03/28/2013	3365488.50	-3341534.93	23953.57	0.699803	16762.77
TOTAL					-176149.58

Main | Curves | Cashflow | Risk | Horizon

Australia 61 2 9777 8600 Brazil 5511 3048 4500 Europe 44 20 7330 7500 Germany 49 69 920410
Hong Kong 852 2977 6000 Japan 81 3 3201 8900 Singapore 65 6212 1000 U.S. 1 212 318 2000 Copyright 2005 Bloomberg L.P.
G566-178-0 18-Nov-05 15:09:39

Figure 1.7 *Source: Reproduced with permission from Bloomberg.*



<HELP> for explanation. P198 Corp SWPM

Options | New Deal | Copy Deal | View | SWAP MANAGER

Deal Counterparty T6qyNWPA12 Ticker / GIC Series 0001 Deal# SL6J0C2T DETAIL

Curve #23 USD Swaps (30/360,S/A)

Current Market			6 Month –50bp			6 Month +0bp			6 Month +50bp		
#	Mty/Term	Rate	#	Mty/Term	Rate	#	Mty/Term	Rate	#	Mty/Term	Rate
1	1 DY	3.50000	1	1 DY	3.00000	1	1 DY	3.50000	1	1 DY	4.00000
2	2 DY	3.80000	2	2 DY	3.30000	2	2 DY	3.80000	2	2 DY	4.30000
3	1 WK	3.81688	3	1 WK	3.31688	3	1 WK	3.81688	3	1 WK	4.31688
4	2 WK	3.82000	4	2 WK	3.32000	4	2 WK	3.82000	4	2 WK	4.32000
5	3 WK	3.86000	5	3 WK	3.36000	5	3 WK	3.86000	5	3 WK	4.36000
6	1 MO	3.94563	6	1 MO	3.44563	6	1 MO	3.94563	6	1 MO	4.44563
7	2 MO	4.02563	7	2 MO	3.52563	7	2 MO	4.02563	7	2 MO	4.52563
8	3 MO	4.14000	8	3 MO	3.64000	8	3 MO	4.14000	8	3 MO	4.64000
9	4 MO	4.20000	9	4 MO	3.70000	9	4 MO	4.20000	9	4 MO	4.70000
10	5 MO	4.26000	10	5 MO	3.76000	10	5 MO	4.26000	10	5 MO	4.46000
11	6 MO	4.32875	11	6 MO	3.82875	11	6 MO	4.32875	11	6 MO	4.82875
12	7 MO	4.37025	12	7 MO	3.87025	12	7 MO	4.37025	12	7 MO	4.87025

Horizon Curve Date	10/12/05		Horizon Curve Date	04/12/06		Horizon Curve Date	04/12/06		Horizon Curve Date	04/12/06	
Horizon Settle Date	10/14/05		Horizon Settle Date	04/18/06		Horizon Settle Date	04/18/06		Horizon Settle Date	04/18/06	
GLOBAL CHANGE FIELDS	----→		From 1	To 34	Shift 0.00	From 1	To 34	Shift 0.00	From 1	To 34	Shift 0.00
Pay Leg PV	-3,305,321.54		Pay Leg PV	-3,331,694.62		Pay Leg PV	-3,308,454.83		Pay Leg PV	-3,305,221.38	
Receive Leg PV	3,128,009.55		Receive Leg PV	3,239,101.26		Receive Leg PV	3,143,640.16		Receive Leg PV	3,051,445.05	
Net PV	-177,311.99		Net PV	-72,593.36		Net PV	-164,814.67		Net PV	-253,776.32	

Main | Curves | Cashflow | Risk | Horizon

Australia 61 2 9777 8600 Brazil 5511 3048 4500 Europe 44 20 7330 7500 Germany 49 69 920410
Hong Kong 852 2977 6000 Japan 81 3 3201 8900 Singapore 65 6212 1000 U.S. 1 212 318 2000 Copyright 2005 Bloomberg L.P.
G566-178-0 12-Oct-05 15:48:54

Figure 1.8 *Source: Reproduced with permission from Bloomberg.*

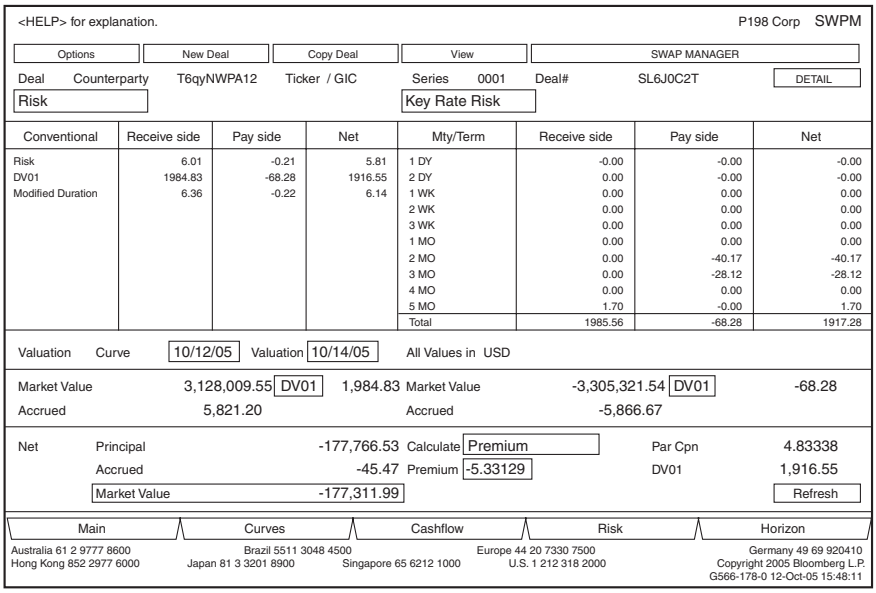


Figure 1.9 *Source: Reproduced with permission from Bloomberg.*

1.12 SWAP VALUATION IN C++

To price a vanilla fixed-for-floating swap in C++, we define a *Swap* class. The *Swap* class is composed of two legs—a *FloatingLeg* class, floatLeg, and a *FixedLeg* class, fixedLeg, which represent the two sides of the swap. Because the calculation of payment dates based on the start date, effective date, and maturity date is required, the *Swap* class utilizes a *Date* class[22] that contains methods for computing date operations. The *Date* class is defined in the "datecl.h" file and will be used throughout this book to compute payment dates.

The *Swap* class contains the following *Date* data members:

```
Date maturity_;              // swap maturity date
Date fixedAccruedDate_;      // fixed interest accrual date
Date floatAccruedDate_;      // floating interest accrual date
Date effectiveDate_;         // effective date
Date settlementDate_;        // settlement date
Date valuationDate_;         // valuation date
```

The *Date* methods are defined in the "datecl.cpp" source file. The *Swap* class is defined with various inline methods to add in the computation of the swap payments, netting, and pricing:

```
double getNotional()         // return notional amount
double calcDV01()            // compute swap DV01
void netPayments()           // net fixed and floating payments
void calcPayDates(Date tradeDate, Date endDate, Date valuation)
```

```
  // calculate pay dates
void setDiscountRates(std::map<double,double> rate)  // set discount rates
```

The *Swap* class utilizes the *floatLeg* and *fixedLeg* data class members to perform many of the operations in these methods. The *Swap* class contains overloaded constructors to receive and store important data for pricing: the maturity date, effective date, settlement date, valuation date, floating (libor) rates, discount rates, fixed swap rate, and valuation type (receive fixed-pay floating or receive floating-pay fixed). The *Swap* class definition is:

SWAP.h

```
#ifndef _SWAP_H__
#define _SWAP_H__

#include "TNT\TNT.h"
#include "datecl.h"
#include <string>
#include <vector>
#include <map>
#define NUM_DATES        100
#define THIRTY            30
#define THREE_SIXTY      360
#define THREE_SIXTY_FIVE 365
#define NOTIONAL         1000000

static std::vector<Date> payDates_;

static double interpolate(double rate1, double rate2, double t1, double t2,
  double x) {
  double dy = rate2 - rate1;
  double dt = t2 - t1;
  double slope = dy/dt;

  return rate1 + slope*x;
}

class FloatingLeg
{
  public:
    FloatingLeg(std::map<double,double> floatLegRate, double floatLegBasis,
      int payFrequency)
                : floatLegRate_(floatLegRate),floatLegBasis_(floatLegBasis),
                  payFrequency_(payFrequency) {}
    FloatingLeg() {}
    virtual ~FloatingLeg() {}
    inline double calcDuration() {
      double duration = 0.0;
      duration = (double) (payDates_[0] -
        valuationDate_ + 1)/THREE_SIXTY_FIVE;
      return duration;
    }
    inline void setEffectiveDate(Date date) { startDate_ = date; }
    inline void setValuationDate(Date date) { valuationDate_ = date; }
    inline void setNotional(double notional) { notional_ = notional; }
    inline Date getEffectiveDate() { return startDate_; }
```

```
inline void setFrequency(int frequency) { payFrequency_ = frequency; }
inline void setMaturityDate(Date mat) { maturityDate_ = mat; }
inline std::vector<double> getPayFloat() { return floatRates; }
inline void setFloatValue(double value) { value_ = value; }
inline double getFloatValue() {
    return value_;
}
inline void setFloatRate(std::map<double,double> rate) {
      floatLegRate_ = rate;
}
inline double calcDV01() {

      double duration = calcDuration();
      double val = getFloatValue();
      double DV = 0.0;

      DV = -(duration*notional_)*((double)1/10000);
      cout << "float DV01 = " << DV << endl;

      return -DV;
}
inline double calcModifiedDuration() {

      double val = calcDV01();
      double marketValue = getFloatValue();
      double MD = (val/(notional_ + marketValue))*10000;

      cout << "float modified duration = " << MD << endl;

      return MD;
}
inline void calcPayFloat() {

      std::vector<Date>::iterator iter;
      double val = 0;
      double diff = 0.0;
      int diff1 = 0.0;
      int d = 0.0;
      int d1 = 0.0;
      Date dateDiff;
      int cnt = 0;
      double floatVal = 0.0;

      for (iter = payDates_.begin(); iter != payDates_.end(); iter++)
      {
        d = payDates_[cnt+1] - payDates_[0] + 1;
        d1 = payDates_[cnt+1] - payDates_[cnt] + 1;

        diff = payDates_[cnt+1] - payDates_[0]+1;
        diff = (double) diff/THREE_SIXTY_FIVE;
        floatVal = interpolate(floatLegRate_[floor(diff)],
          floatLegRate_[ceil(diff)],floor(diff),ceil(diff),diff);

        if (payFrequency_ == 1)
              val = notional_*(THIRTY/THREE_SIXTY)*floatVal;
        else
              val = notional_*((double)d1/THREE_SIXTY_FIVE)*floatVal;
```

```
            floatRates.push_back(val);
            cnt++;
          }
    }
  private:
    double floatLegBasis_;
        std::vector<double> floatRates;
        std::map<double,double> floatLegRate_;
        std::map<double,double> payfloatLeg_;
        Date startDate_;
        Date maturityDate_;
        Date valuationDate_;
        double value_;
        double spread_;
        double notional_;
        double duration_;
        double accrual_;
        int payFrequency_;
};

class FixedLeg
{
  public:
        FixedLeg(double payfixedLeg, double fixedLegRate,
          double fixedLegBasis, int payFrequency)
      : payfixedLeg_(payfixedLeg), fixedLegRate_(fixedLegRate),
        fixedLegBasis_(fixedLegBasis),
                payFrequency_(payFrequency) {}
        FixedLeg() {}
        virtual ~FixedLeg() {}
        inline double calcDV01() {

          double duration = calcDuration();
          double val = getFixedValue();
          double DV = 0.0;

          DV = (duration*notional_)*((double)1/10000);
          cout << "fixed DV01 = " << DV << endl;

          return DV;
        }
        inline void setEffectiveDate(Date date) { effectiveDate_ = date; }
        inline double calcDuration() {

          double sum = 0;
          double val = getFixedValue();
          double duration = 0.0;
          double dis = 0.0;

          for (int i = 0; i < payfixedLeg_.size(); i++)
             sum = sum + payfixedLeg_[i]*((double)(payDates_[i+1] -
               valuationDate_ + 1)/
                (maturityDate_ - valuationDate_ + 1));

             sum = sum + notional_;
             duration = sum/val;
```

```
      cout << "fixed duration = " << duration << endl;

      return duration;
}
inline double calcModifiedDuration() {

   double val = calcDV01();
   double marketValue = getFixedValue();
   double MD = (val/(notional_ + marketValue))*10000;

   return MD;
}
inline void setFixedValue(double val) { value_ = val; }
inline void setValuationDate(Date date) { valuationDate_ = date; }
inline double getFixedValue() {
   return value_;
}
inline void setNotional(double notional) { notional_ = notional; }
inline void setFrequency(int frequency) { payFrequency_ =
  frequency; }
inline void setFixedRate(double rate) { fixedLegRate_ = rate; }
inline void setMaturityDate(Date mat) { maturityDate_ = mat; }
std::vector<double> getPayFixed() { return payfixedLeg_; }
inline void calcPayFixed()
{
  std::vector<Date>::iterator iter;
  double val = 0;
  int diff = 0;
  Date dateDiff;
  int cnt = 0;

  for (iter = payDates_.begin(); iter != payDates_.end(); iter++)
  {
    if (payFrequency_ == 1)
    {
          if (cnt <= payDates_.size())
          {
                if (cnt + payFrequency_ < payDates_.size())
                   diff = payDates_[cnt+payFrequency_] -
                     payDates_[cnt]+1;
                else
                   diff = 0;

                if ((cnt != 0) && (cnt % payFrequency_ == 0))
                    val = notional_*(THIRTY/THREE_SIXTY)*
                      fixedLegRate_;
                else
                    val = 0;
           }
           else
              val = notional_*(THIRTY/THREE_SIXTY)*fixedLegRate_;
     }
     else
     {
           if (cnt <= payDates_.size())
               {
             if (cnt + payFrequency_ <= payDates_.size())
```

```
                        {
                      // subtract five because there are 5 less days in a
                      // 360 day year
                          diff = (payDates_[cnt+payFrequency_] -
                            payDates_[cnt] + 1) - 5;
                        }
                    else
                      diff = 0;

                    if ((cnt > 0) && ((cnt-1) % payFrequency_ == 0))
                      val = notional_*((double)diff/THREE_SIXTY)*
                        fixedLegRate_;
                    else
                      val = 0;
                      }
                      else
                      {
                        val = 0;
                      }
            }
            cnt++;
            payfixedLeg_.push_back(val);
          } // for
        }
  private:
        std::vector<double> payfixedLeg_;
        double fixedLegRate_;
        double fixedLegBasis_;
        double duration_;
        double value_;
        double accrual_;
        double notional_;
        double basis_;
    int payFrequency_;
        Date effectiveDate_;
        Date maturityDate_;
        Date valuationDate_;
};

class Swap
{
  public:
        Swap() : notional_(NOTIONAL), maturity_("12/31/2010"),
          swapType(0) {}
        Swap(double notional, Date maturity, Date effectiveDate,
          Date settlementDate, Date valuation,
        std::map<double,double> liborRate, std::map<double,double> disc,
          double fixedRate, int type)
           : notional_(notional), maturity_(maturity),
            effectiveDate_(effectiveDate), settlementDate_(settlementDate),
              valuationDate_(valuation), floatRates_(liborRate),
              fixedRate_(fixedRate), swapType(type)
        {
          getNotional();
          calcPayDates(effectiveDate_,maturity_,valuationDate_);
          fixedLeg.setNotional(notional_);
          fixedLeg.setValuationDate(valuationDate_);
```

```
  fixedLeg.setFrequency(2);
  fixedLeg.setFixedRate(fixedRate);
  fixedLeg.setMaturityDate(maturity);
  fixedLeg.calcPayFixed();
  fixedLeg.setEffectiveDate(effectiveDate_);
  floatLeg.setMaturityDate(maturity);
  floatLeg.setFrequency(4);
  floatLeg.setNotional(notional_);
  floatLeg.setFloatRate(liborRate);
  floatLeg.setValuationDate(valuationDate_);
  setDiscountRates(disc);
  floatLeg.calcPayFloat();
  netPayments();
  calcDV01();
}
virtual ~Swap() {}
inline double getNotional() {
       return notional_;
}
inline void setDiscountRates(std::map<double,double> rate) {
  discRates_ = rate;
}
inline double calcDV01() {

  double val = 0;

  if (swapType == 0)
       val = fixedLeg.calcDV01() - floatLeg.calcDV01();
  else
       val = floatLeg.calcDV01() - fixedLeg.calcDV01();

  cout << "Swap DV01 = " << val << endl << endl;

  return val;
}
inline void calcPayDates(Date tradeDate, Date endDate,
  Date valuation)
{
  effectiveDate_ = tradeDate-1;

  if (effectiveDate_ == Date::SATURDAY)
       effectiveDate_ = effectiveDate_ + 2;
  else if (effectiveDate_ == Date::SUNDAY)
       effectiveDate_ = effectiveDate_ + 1;

  Date currDate = effectiveDate_;
    int cnt = 0;

  while (currDate <= endDate)
  {
       currDate.AddMonths(3);
       while (currDate.day > effectiveDate_.day)
    currDate = currDate - 1;

       if (currDate <= valuation)
       {
          if (currDate.day_of_week == Date::SATURDAY)
```

```
                    currDate = currDate + 2;
                else if (currDate.day_of_week == Date::SUNDAY)
                    currDate = currDate + 1;
                else if (currDate ==
                  currDate.ChristmasDay(currDate.year))
                    currDate = currDate + 1;

                fixedAccruedDate_ = currDate;
            }
            if ((currDate <= endDate) && (currDate >= valuation))
            {
                if (currDate.day_of_week == Date::SATURDAY)
                    currDate = currDate + 2;
                else if (currDate.day_of_week == Date::SUNDAY)
                    currDate = currDate + 1;
                else if (currDate ==
                  currDate.ChristmasDay(currDate.year))
                    currDate = currDate + 1;

                payDates_.push_back(currDate);
                cnt++;
             }
    }
  }
  inline void netPayments()
  {
    double val = 0.0;
    double y = 0.0;
    std::vector<double> fixed = fixedLeg.getPayFixed();
    std::vector<double> fl = floatLeg.getPayFloat();
    double x = 0;
    double sum = 0.0;
    double sumfix = 0.0;
    double sumfloat = 0.0;

    if (swapType == 0)
    {
      fixedAccrued_ =  notional_*fixedRate_*((valuationDate_ -
        fixedAccruedDate_))/THREE_SIXTY;
          floatAccrued_ = -notional_*floatRates_[0]*
            ((valuationDate_ - fixedAccruedDate_))/THREE_SIXTY_FIVE;
    }
    else
    {
          fixedAccrued_ = -notional_*fixedRate_*((valuationDate_ -
            fixedAccruedDate_))/THREE_SIXTY;
          floatAccrued_ =  notional_*floatRates_[0]*
            ((valuationDate_ - fixedAccruedDate_))/THREE_SIXTY_FIVE;
    }

    int cnt = 0;
    for (int i = 0; i < payDates_.size(); i++)
    {
          x = (double) (payDates_[i+1] - payDates_[0] +
            1)/THREE_SIXTY_FIVE;
          y = interpolate(discRates_[floor(x)],discRates_[ceil(x)],
            floor(x),ceil(x),x);
```

```
                if (swapType == 0)
                {
                  val = (fixed[i] - fl[i])*y;
                  sumfix = sumfix + fixed[i]*y;
                  sumfloat = sumfloat - fl[i]*y;
                }
                else
                {
                  val = (fl[i] - fixed[i])*y;
                  sumfix = sumfix - fixed[i]*y;
                  sumfloat = sumfloat + fl[i]*y;
                }
                sum = sum + val;
          }
          fixedLeg.setFixedValue(sumfix);
          floatLeg.setFloatValue(sumfloat);

          cout << "Fixed Accrued = " << fixedAccrued_ << endl;
          cout << "Float Accrued = " << floatAccrued_ << endl;
          cout << "Accrued = " << fixedAccrued_ + floatAccrued_ << endl;
          cout << "Principal = " << sum << endl;
          cout << "Market Value = " << sum + (fixedAccrued_ + floatAccrued_)
            << endl;
        }
  private:
        int swapType;
        double notional_;
        double fixedAccrued_;
        double floatAccrued_;
        double fixedRate_;
        FloatingLeg floatLeg;
        FixedLeg fixedLeg;
        Date maturity_;
        Date fixedAccruedDate_;
        Date floatAccruedDate_;
        Date effectiveDate_;
        Date settlementDate_;
        Date valuationDate_;
        std::string index;
        std::map<double,double> discRates_;
        std::map<double,double> floatRates_;
        double value;
};

#endif _SWAP_H__
```

Consider the preceding swap with the following characteristics:

```
Notional = $3,300,000
Maturity = March 28, 2013
Effective Date = March 29, 2004
Valuation Date = October 14, 2005
Swap Rate = 3.969%
Floating Rate = 3 Mo. T-Bill
Basis Spread = 0.00
```

We read the following (Table 1.19) in the 3-Mo T-Bill data from a file (taken from Bloomberg).

Table 1.19

Maturity	3-Mo T-Bill	Discount	Maturity	3-Mo T-Bill	Discount
1 DY	0.0400000	0.999708	9 MO	0.0445188	0.967342
2 DY 0000	0.0380000	0.999708	10 MO	0.0448525	0.963507
1 WK	0.0381688	0.999258	11 MO	0.0451725	0.959660
2 WK	0.0382000	0.999258	1 YR	0.0454563	0.955712
3 WK	0.0386000	0.998517	2 YR	0.0463900	0.912185
1 MO	0.0394583	0.996614	3 YR	0.0471200	0.869450
2 MO	0.0402583	0.993225	4 YR	0.0472200	0.829536
3 MO	0.0414000	0.989193	5 YR	0.0475900	0.790131
4 MO	0.0420000	0.985853	6 YR	0.0479100	0.752230
5 MO	0.0426000	0.982445	7 YR	0.0481800	0.715816
6 MO	0.0432875	0.978239	8 YR	0.0486000	0.679864
7 MO	0.0437025	0.974794	9 YR	0.0487000	0.647234
8 MO	0.0441225	0.971079	10 YR	0.0490200	0.614542

To discount the cash flows on the scheduled payment dates, the floating (3-Mo T-Bill) and discount rates are linearly interpolated using the following globally defined function:

```
static double interpolate(double rate1, double rate2, double t1, double
t2, double x)
{
    double dy = rate2 - rate1;
    double dt = t2 - t1;
    double slope = dy/dt;

    return rate1 + slope*x;

}
```

We interpolate using the **floor** and **ceil** built-in math routines of the rate we want to interpolate. For instance, in the `FloatLeg` function `calcPayFloat`, we call

```
floatVal = interpolate(floatLegRate_[floor(diff)],floatLegRate_[ceil(diff)],
  floor(diff),ceil(diff),diff);
```

It is important to note that this is a simple form of interpolation. In actual practice to get more accurate interpolated values, we would want to bootstrap the yield curve using liquid instruments like T-bill futures and Eurodollar futures or use a numerical technique like cubic splines.

The main function is

MAIN.cpp

```
#include <strstrea.h>
#include <fstream.h>
```

```
#include <stdlib.h>
#include <iostream.h>
#include <string.h>
#include <math.h>
#include <map>
#include "Swap.h"
#define SIZE_X 100

void main()
{
  cout.setf(ios::showpoint);
  cout.precision(8);

  cout << "Swap Pricing Pay Fixed  " << endl << endl;
  Date start = "3/29/2004";
  cout << "Start date = " << start << endl;
  Date maturity = "3/28/2013";
  cout << "Maturity = " << maturity << endl;
  Date valuation = "10/14/2005"; //today";
  cout << "Valuation = " << valuation << endl;
  Date effectiveDate = "today";
  double notional = 3300000;
  cout << "Notional = " << notional << endl;
  double swapRate = 0.03969;
  cout << "Swap Rate = " << swapRate << endl;

  std::vector<double> mat;
  std::map<double,double> libor;
  std::map<double,double> discRate;
  char buffer[SIZE_X];
  char dataBuffer[SIZE_X];
  char* str = NULL;
  double yr = 0.0;
  double rate = 0.0;
  int swapType = 1;  // receive fixed-pay float ; 1 = pay fixed-receive
                     // float


  const char* file = "c:\\swapData.txt";
  ifstream fin;              // input file stream
  fin.clear();
  fin.open(file);

  if (fin.good())
  {
      while (!fin.eof())
      {
          fin.getline(buffer,sizeof(buffer)/sizeof(buffer[0]));
          //cout << buffer << endl;
          istrstream str(buffer);
          // Get data
          str >> dataBuffer;
          yr = atof(dataBuffer);

          str >> dataBuffer;
          if (strcmp(dataBuffer,"MO") == 0)
              yr = (double) yr/12;
```

```
          else if (strcmp(dataBuffer,"WK") == 0)
              yr = (double) yr/52;
          else if (strcmp(dataBuffer,"DY") == 0)
              yr = (double) yr/365;
          mat.push_back(yr);

          str >> dataBuffer;
          rate = atof(dataBuffer);
          libor[yr] = rate;

          str >> dataBuffer;
          rate = atof(dataBuffer);
          discRate[yr] = rate;
      }
   }
   else
      cout << "File not good!" << "\n";

   fin.close();

   Swap s(notional,maturity,start,start+1,valuation,libor,discRate,
     swapRate,swapType);
}
```

We get the following results:

```
Fixed Accrued = 5821.2000
Float Accrued = -5867.3096
Accrued = -46.109589
Principal = -176178.34
Market Value = -176224.45
```

The risk measures are as follows:

```
fixed duration = 5.8713815
fixed DV01 = 1937.5559
float duration = 0.20821918
float DV01 = -68.712329
Swap DV01 = 1868.8436
```

1.13 Bermudan Swaption Pricing in Matlab

Interest rate swaps have the characteristics of futures contracts. As such, they are used to lock in future interest rate positions, usually for longer periods than can be obtained with exchange-traded futures. Financial managers, though, who want downside protection for their position with the potential for gains if conditions become favorable, can also take a position in *swaptions* or options on swaps. Swaptions give the holder the right, but not the obligation, to enter into a swap at maturity: for example, a fixed-rate payer's position (or a floating-rate payer's position), with the exercise price set by the fixed rate on the swap. Bermudan swaptions, however, give the holder the right to enter on discrete prespecified

dates throughout the life of the swaption. The following Matlab implementation[23] values a Bermudan swaption:

bermudan_swaption.m

```
function []=bermudan_swaption()

%This program can work for arbitrary no. of factors. You have to specify no.
%of factors as well as volatility structure for each factor. The volatility
%structure can be obtained from principal component analysis of correlation
%matrix and adjusting to calibrated volatilities as done in excellent paper
%by Rebonato. See my web page for the references
%(http://www.geocities.com/anan2999). It does not take correlation
%structure as input. You can also specify CEV constant alpha for skew.
%Remember changing this constant changes effective volatility.

%randn('state',[1541045451;4027226640]) % add a good random number seed
%here if you wish.
%if you don't matlab will choose its own seed.

delta=.25; %Tenor spacing. usually .25 or .5

P=5000;  % No. of paths, do not try more than 5000 paths unless you are
         % very patient
T_e1=6.0;               %maturity of underlying swap in years(must be an
                        %exact multiple of delta)
T_x1=5.75;              %last exercise date of the swaption (must be an
                        %exact multiple of delta)
T_s1=3.0;               %lockout date (must be an exact multiple of delta)


T_e=T_e1/delta+1;
T_x=T_x1/delta+1;
T_s=T_s1/delta+1;
N=T_e;

F=2;   % number of factors. If you change this line also change volatility
       % structure appropriately
alpha=1.0;%CEV constant alpha for skew.Remember changing this value changes
          %effective volatility
%It is 1.0 for lognormal model.
k=.1; % strike, fixed coupon
pr_flag=+1; %payer receiver flag; assumes value of +1 for a payer swaption
%and a value of -1 for a receiver swaption.


n_spot=2;
L=repmat(.10,[P,T_e+1]);
vol=repmat(0,[T_e,F]);
for n=1:N,
   for f=1:F,
      if(f==1)
         vol(n,f)=.15; %volatility of first factor
      end
      if(f==2)
         vol(n,f)= (.15-(.009*(n)*.25).^.5); %volatility of second factor
```

```
      end
   end
end
%You can add more volatility factors in the above line but please also
%change F accordingly
%drift=repmat(0,[P,F]);
money_market=repmat(1,[T_x,P]);
swap=repmat(0,[T_x,P]);
B=repmat(1,[P,T_e]);


money_market(2,:)=money_market(1,:).*(1+delta*L(:,1))';
increment=repmat(0,[P,1]);
drift=repmat(0,[P,F]);

for t= 2 : T_x,

   t


  normal_matrix=randn([P,F]);
  drift(:,:)=0;
   for n= t : T_e,
      increment(:,1)=0;

                %      n
      for f=1:F,

         drift(:,f)=drift(:,f)+ delta*vol(n-n_spot+1,f).*
                     ((L(:,n).^alpha)./(1+delta.*L(:,n))); %

         increment(:,1)=increment(:,1)+vol(n-n_spot+1,f).*
                          (L(:,n).^alpha)./L(:,n)...
            .*(normal_matrix(:,f).*sqrt(delta)-.5.*vol(n-n_spot+1,f).*
              (L(:,n).^alpha)./L(:,n)...
            .*delta+drift(:,f).*delta);
      end

      L(:,n)=L(:,n).*exp(increment(:,1));
      L(L(:,n)<.00001,n)=.00001;
   end

   B(:,t)=1.0;
   for n=t+1:T_e,
      B(:,n)=B(:,n-1)./(1+delta.*L(:,n-1));
   end



   money_market(t+1,:)=money_market(t,:).*(1+delta*L(:,n_spot))';

   if((t>= T_s) & (t <=T_x))
      for n=t:(T_e-1), %//the swap leg is determined one date before
                       %//the end

                  swap(t,:)=swap(t,:)+  (B(:,n+1).*
                              (L(:,n)-k).*pr_flag*delta)' ;
```

```
      end
   end
   n_spot=n_spot+1;

end



value=repmat(0,[P,1]);
stop_rule=repmat(T_x,[P,1]);

value(swap(T_x,:)>0,1) = (swap(T_x,swap(T_x,:)>0))';
coeff=repmat(0,[T_x,6]);

for t=(T_x-1):-1:T_s,
   i=0;
   a=0;
   y=0;
   for p=1:P,
      if (swap(t,p)> 0.0)
         i=i+1;
         a(i,1)=1;
         a(i,2)=swap(t,p);
         a(i,3)=swap(t,p)*swap(t,p);
         a(i,4)=money_market(t,p);
         a(i,5)=money_market(t,p)*money_market(t,p);
         a(i,6)=money_market(t,p)*swap(t,p);

         y(i,1)= money_market(t,p)/money_market(stop_rule(p,1),p) *
                   value(p,1);

      end

   end



   temp=inv(a'*a)*(a'*y);
   coeff(t,:)=temp';

   expec_cont_value=repmat(0,[P,1]);
   exer_value=repmat(0,[P,1]);

   expec_cont_value(:,1)=(coeff(t,1)+coeff(t,2).*swap(t,:)+
                          coeff(t,3).*swap(t,:)...
      .*swap(t,:)+coeff(t,4).*money_market(t,:)+
        coeff(t,5).*money_market(t,:)...
      .*money_market(t,:)+coeff(t,6).*money_market(t,:).*swap(t,:))';


exer_value(swap(t,:)>0,1)=(swap(t,swap(t,:)>0))';


 value((exer_value(:,1)>expec_cont_value(:,1))&(swap(t,:)>0)',1)...
    =exer_value((exer_value(:,1)> expec_cont_value(:,1))&(swap(t,:)>0)',1);

 stop_rule((exer_value(:,1)>expec_cont_value(:,1))&(swap(t,:)>0)',1)=t;
```

```
end

   price=0;
   for p=1:P,
      price=price+ (value(p,1)/(money_market(stop_rule(p,1),p)))/P;

   end

   price
```

ENDNOTES

1. This is the case for a plain vanilla fixed-for-floating. A hedge on a bond portfolio with an amortization of the principal or a hedge on the total return on bond portfolio would require use of constant maturity swaps (or index amortization swaps) and total return swaps, which do not have swap futures equivalents (as of today).
2. A *Eurodollar* is a dollar deposited in a U.S. or foreign bank outside the United States. The Eurodollar interest rate is the rate of interest earned on Eurodollars deposited by one bank with another bank.
3. It can be shown that if an underlying asset of a long futures contract is strongly positively correlated with interest rates, futures prices will be higher than forward prices because futures are settled daily, and the gain can be invested at a higher-than-average rate of interest since the positive correlation will make it more likely rates will increase. Similarly, when the underlying asset is strongly negatively correlated with interest rates, the futures position will incur an immediate loss, and this loss will tend to be financed at a lower-than-average rate of interest. An investor with a long position in a forward contract rather than a futures contract is not affected in this way by rate movements.
4. Hull, J. (1997), *99*.
5. Id. 99.
6. Id. 100.
7. Id. 100.
8. Id. 100.
9. Hull, J. (1997), 97.
10. See *http://www.academ.xu.edu/johnson/*.
11. In actuality, the manager could purchase only five contracts because she can only purchase an integer multiple of contracts. Thus, she would be underhedged by 0.115 T-bill units, but if she bought six contracts, she would be overhedged by 0.985 T-bill units.
12. We can also compute the duration using a continuously compounded yield, y_i, $1 \leq i \leq n$, as

$$D = \sum_{i=1}^{n} t_i \left[\frac{c_i e^{-y_i t_i}}{B} \right]$$

where $B = \sum_{i=1}^{n} c_i e^{-y_i t_i}$.

13. The term "yield modified duration" refers to the traditional analytic formulation for modified duration using a flat discount rate.
14. To compute DV01, shift the yield curve up by one basis point, recompute the duration using yield + 1bp, and then subtract the initial duration before the one basis point shift.
15. Reproduced with permission from "Hedging a Fixed-Income Portfolio with Swap Futures," CBOT Interest Rate Swap Complex White Paper.
16. Audley, D., Chin, R, and Ramamuthy, S., "Term Structure Modeling" in *Interest Rate, Term Structure, and Valuation Modeling*, edited by Fabozzi, F., Wiley (2002)., pg. 95.
17. Id., pg. 95.
18. Typically, $n = 2$ as most bonds use semiannual compounding.
19. A cubic "spline" is a piecewise polynomial function, made up of individual polynomial sections or segments that are joined together at (user-selected) points known as *knot points*. A cubic spline is a function of order three, and a piecewise cubic polynomial that is twice differentiable at each knot point. At each knot point, the slope and curvature of the curve on either side must match. An exponential cubic function would fit an exponential curve through the discount points. Thus, cubic and exponential splines are used to fit a smooth curve to bond prices (yields) given the term discount factors.

    See James and Webber (2000), *Interest Rate Modeling*, Wiley, pp. 430-432; Waggoner, D. (1997); Pienaar, R. and Choudhry., M. article in Fabozzi (2002), "Fitting the Term Structure of Interest Rates Using the Cubic Spline Methodology," pg. 157-185; O. De la Grandville (2001), *Bond Pricing and Portfolio Analysis*, MIT Press, pp. 248-252; Vasicek, O. and Fong, H. (1982), "Term Structure Modeling Using Exponential Splines," *Journal of Finance* 37, 1982, pp. 339-361.
20. Developed by Stefano Galiani (2003).
21. See MATLAB Fixed-Income Toolkit User's Guide (2002), The MathWorks.
22. The *Date* class is an open source library written by James M. Curran (1994).
23. Written by Ahsan Amin. See *http://www.geocities.com/anan2999/.*