CHAPTER 3

MORTGAGE-BACKED SECURITIES

SECTIONS

Mortgage-backed securities (MBSs) and mortgage pass-throughs (PT) are claims on a portfolio of mortgages. MBSs are created when a federal agency, mortgage banker, bank, or investment company buys up mortgages of a certain type—i.e., FHA (Federal Home Administration) or VA (Veterans' Administration) insured—and then sells claims on the cash flows from the portfolio as MBSs, with the proceeds of the MBS sale being used to finance the purchase of the mortgages. There are two types of MBS: agency and conventional (private-label).[1]

Agency MBSs, such as a GNMA pass-through, are securities with claims on a portfolio of mortgages insured against default risk by FHM, VA, or FmHA (Farmers Mortgage Home Administration). A mortgage banker, bank, or investment company presents a pool of FHA, VA, or FmHA mortgages of a certain type (30-year fixed, 15-year variable rate, etc.) to GNMA (Ginnie Mae). If the mortgage pool is in order, GNMA will issue a separate guarantee that allows the MBSs on the mortgage pool to be issued as a GNMA PT. Other agency MBSs include the Federal Home Loan Mortgage Corporation (FHLMC) MBSs, which are claims on a portfolio of conventional mortgages. The FHLMC issues agency MBSs, whereby the FHLMC buys mortgages from the mortgage originator, and then creates an MBS referred to as a *participation certificate*, which it issues through a network

of dealers. FHLMC has a swap program whereby FHLMC swaps MBSs for a savings and loan's or commercial bank's portfolio of mortgages of a certain type. Other government agencies such as FNMA (Fannie Mae) issue several types of MBS: participation certificates, swaps, and PTs. With these certificates, homeowners' mortgage payments pass from the originating bank through the issuing agency to the holds of the certificates.

Conventional types, also known as private-label types, are issued by commercial banks (via their holding companies), S&Ls, mortgage bankers, and investment companies. Conventional issued MBSs include those issued by Prudential Home, Chase Mortgage, Citi-Corp Housing, Ryland/Saxon, GE Capital, and Countrywide. Conventional PTs must be registered with the SEC. These PTs are often insured with external insurance in the form of a letter of credit (LOC) of the private-label issuer, as well as internal insurance through the creation of senior and junior classes of the PT structured by the private-label issuer.

There is both a primary and a secondary market for MBS. In the primary market, investors buy MBSs issued by agencies or private-label investment companies either directly or through dealers. Many of the investors are institutional investors. Thus, the creation of MBS has provided a tool for having real estate financed more by institutions. In the primary market, MBS issue denominations are typically between $25,000 to $250,000 (with some as high as $1M) and some have callable features. In the secondary market, MBSs are traded over-the-counter (OTC). OTC dealers are members of the Mortgage-Backed Securities Dealer Association (MSDA).

MBSs are some of the most complex securities to model and value due to their sensitivity to prepayment and interest rates, which affects the timing, frequency, and size of cash flows to investors. Cash flows (CFs) from MBSs are the monthly CFs from the portfolio of mortgages (referred to as the *collateral*). Cash flows include interest on principal, scheduled principal, and prepaid principal. Cash flow analysis is essential in the valuation of any MBS given their impact by the underlying features of the MBS, including *weighted average maturity* (WAM), *weighted average coupon rate* (WAC), *pass-through rate* (PT rate), and *prepayment rate* or *speed*. The WAM is effectively the duration, or weighted length of time, of all the payment of MBS cash flows to be paid out to investors. The WAC is the rate on a portfolio of mortgages (collateral) that is applied to determine scheduled principal. The PT rate is the interest on principal and is lower than the WAC, with the difference going to the MBS issuer. The prepayment rate or speed is the assumed prepayment rate made by homeowners of mortgages in the pool.

In this chapter, we discuss MBS pricing and modeling in detail. In §3.1, we discuss prepayment and PSA models for MBS pricing. In §3.2, we give numerical examples using Excel of how the prepayment models work. In §3.3, we discuss MBS pricing, quoting, and the value and return to investors based on different prepayment and interest rate assumptions. In §3.4, we discuss prepayment risk and the average life of MBS. In §3.5, we review in detail a numerical implementation in C++ and Excel for valuation and cash flow analysis of MBS using Monte Carlo simulation. In §3.6, we give numerical examples using the Fixed-Income Toolbox in Matlab. In §3.7, we discuss MBS derivatives, including collateralized mortgage obligations (CMOs) and sequential-pay tranche structures. We give examples using Excel. In §3.8, we give an implementation of a CMO in C++. In §3.9, we discuss planned amortization classes (PAC) and their structures. In §3.10, we review stripped MBSs, including interest-only (IO) and principal-only (PO) securities. In §3.11,

we discuss interest rate risk of MBSs. Finally, in §3.12, we discuss hedging MBSs and using MBSs for balance sheet asset-liability management.

3.1 PREPAYMENT MODELS

MBS valuation models typically assume a prepayment rate or speed. Investors and issuers apply different prepayment models in analyzing MBS. Most models, though, are compared to a benchmark model or rate. The benchmark model is the one provided by the Public Securities Association (PSA). PSA measures speed by the Conditional Prepayment Rate (CPR). CPR is the proportion of the remaining mortgage balance that is prepaid each month and is quoted on an annual basis. The monthly rate is referred to as the Single-Monthly Mortality rate (SMM) and is given by:

$$SMM = 1 - (1 - CPR)^{1/12} \tag{3.1}$$

The estimated monthly prepayment is:

$$\textit{Monthly prepayment} = SMM \cdot [\textit{Beginning of month balance} - \textit{Sched. prin. for month}]$$

For example, if CPR $= 6\%$, beginning-of-the-month balance $= \$100$M, and scheduled principal for month $= \$3$M, then the estimated prepaid principal for the month would be $\$0.499$M:

$$SMM = 1 - [1 - .06]^{1/12} = .005143$$
$$\textit{Monthly prepaid principal} = .005143[\$100M - \$3M] = \$0.499M$$

In the PSA model, CPR depends on the maturity of the mortgages. PSA's standard model assumes that for a 30-year mortgage (360 months), the CPR is equal to .2% the first month, grows at that rate for 30 months to equal 6%, and stays at 6% for the rest of the mortgage's life. This model is referred to as the 100% PSA model. Figure 3.1 shows the prepayment rate as a function of time in months.

The estimation of CPR for month t is:

$$CPR = \begin{cases} 0.06\left(\frac{t}{30}\right), & \text{if } t \leq 30 \\ 0.06, & \text{if } t > 30 \end{cases} \tag{3.2}$$

As an example, the CPR for month five is:

$$CPR = .06\left(\frac{5}{30}\right) = .01$$
$$SMM = 1 - [1 - .01]^{1/12} = .000837$$

PSA's model can be defined in terms of different speeds by expressing the standard model (100% PSA) in terms of a higher or lower percentage, such as 150% or 50%. In a period of lower rates, the PSA model could be 150%, and in a period of higher rates, it

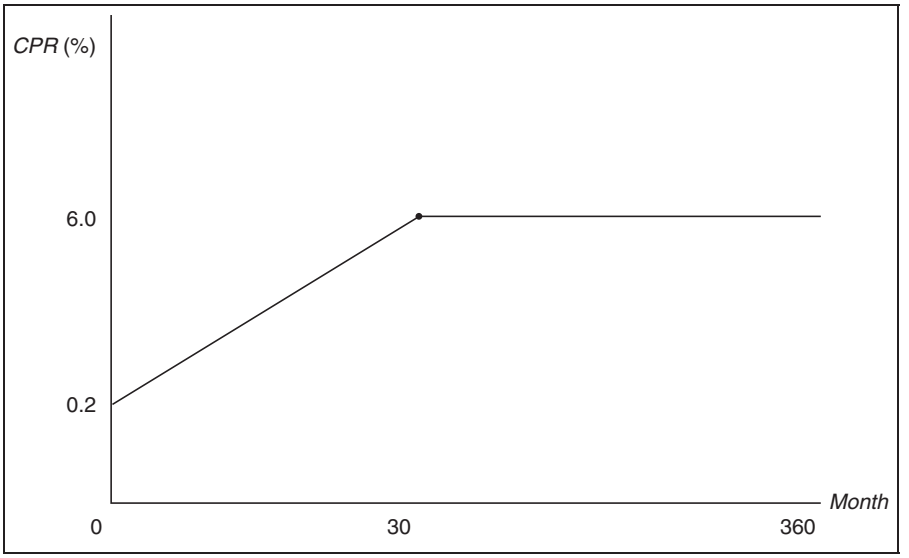

Figure 3.1 100% PSA Model.

could be 50%. For the 100% PSA model, the average time a 30-year mortgage is held is 17 years; for a 225% PSA model, it is 8 years. Figure 3.2 shows the different prepayment rates as a function of time in months.

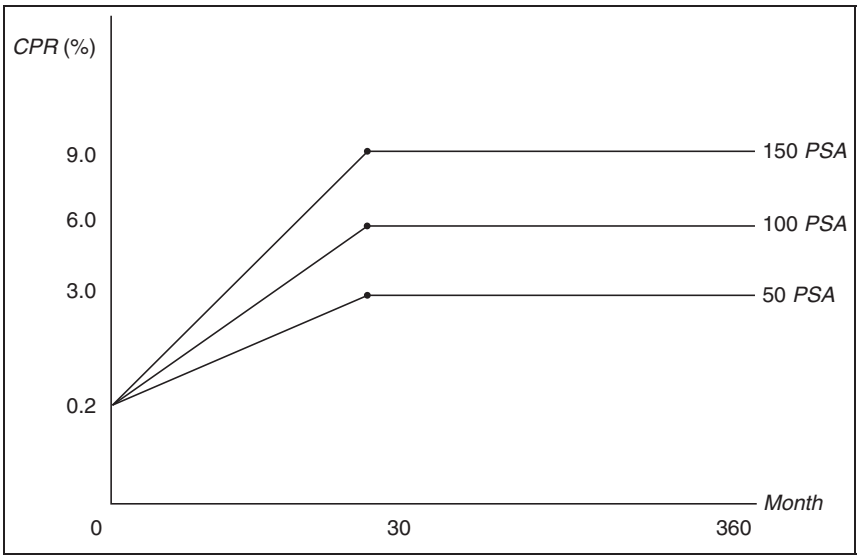

Figure 3.2 PSA Models.

Suppose we want to compute the CPR and SMM for month five with 165 PSA speed. Then we compute the following based on (3.1) and (3.2):

$$\begin{aligned} \textit{CPR} &= .06\left(\frac{5}{30}\right) = .01 \\ 165\textit{CPR} &= 1.65(.01) = .0165 \\ \textit{SMM} &= 1 - [1 - .0165]^{1/12} = .0001386 \end{aligned}$$

3.2 Numerical Example of Prepayment Model

Let $p =$ monthly scheduled mortgage payment, $F_0 =$ the face value of the underlying mortgage pool of the MBS, $M = \textit{WAM} =$ weighted average of the number of months remaining until maturity, $I =$ interest rate payment, $\textit{SP} =$ scheduled principal payment, $\textit{PP} =$ prepaid principal, $R_A =$ annual interest rate (WAC), $B_i, i = 1, \ldots, 360$, the remaining mortgage balance in month i, and $\textit{CF}_i, i = 1, \ldots, 360$, the cash flow in the ith month. Note that the balance in month 1 is the initial face value of the MBS pool, $B_1 = F_0$. The following formula gives p:

$$p = \frac{F_0}{\left(\frac{1-1/(1+(R_A/12))^M}{R_A/12}\right)} \tag{3.3}$$

Consider a mortgage portfolio with an underlying face value of \$100M, a WAC $= 9\%$, a WAM $= 360$ months, and a prepayment speed $= 100\%$ PSA. We need to compute the various cash flows for the first month. The first monthly principal payment is as follows:

$$p = \frac{\$100M}{\left(\frac{1-1/(1+(.09/12))^{360}}{.09/12}\right)} = \$804,600$$

The interest payment is:

$$I = \left(\frac{.09}{12}\right)\$100M = \$750,000$$

The scheduled principal payment is:

$$\textit{SP} = \$804,600 - \$750,000 = \$54,600$$

The estimated prepaid principal in the first month is:

$$\begin{aligned} \textit{CPR} &= \left(\frac{1}{30}\right).06 = .002 \\ \textit{SMM} &= 1 - [1 - .002]^{1/12} = .0001668 \\ \textit{PP} &= .0001668[\$100M - \$54,620] = \$16,667 \end{aligned}$$

The first-year cash flow is computed as:

$$\begin{aligned} CF_1 &= p + PP + I \\ &= \$804,600 + \$750,000 + \$16,667 = \$821,295 \end{aligned}$$

The beginning balance for the second month is:

$$\begin{aligned} B_2 &= B_1 - SP - PP \\ &= \$100M - \$54,600 - \$16,667M = \$99.929M \end{aligned}$$

The second-month cash flows are computed as follows. The second monthly payment is:

$$p = \frac{\$99.9287M}{\left[\frac{1-1/(1+(.09/12))^{359}}{.09/12}\right]} = \$804,488$$

The interest payment is:

$$I = \left(\frac{.09}{12}\right)\$99.9287M = \$749,465$$

and the scheduled principal payment is:

$$SP = \$804,488 - \$749,465 = \$55,023$$

The estimated prepaid principal is:

$$\begin{aligned} CPR &= \left(\frac{2}{30}\right).06 = .004 \\ SMM &= 1 - [1 - .004]^{1/12} = .0003339 \\ PP &= .0003330[\$99.9287M - \$55,023] = \$33,352 \end{aligned}$$

Thus, the second-month cash flows are computed as:

$$CF_2 = \$749,400 + \$55,023 + \$33,352 = \$837,840$$

The remaining month cash flows are computed similarly.

Consider now a mortgage portfolio with a face value of \$100M, a WAC $= 8.125\%$, a WAM $= 357$ months, a PT rate $= 7.5\%$, and a prepayment $= 165\%$ PSA. Note that because the WAM is not 360 months, but rather 357 months, the pool age is "seasoned" so that the first month of payments actually starts in month four, and not month one. Moreover, interest payments are calculated using the PT rate. However, scheduled principal and mortgage payments are computed using the WAC rate.

In this example, the schedule monthly mortgage payment is:

$$p = \frac{\$100M}{\left[\frac{1-1/(1+(.08125/12))^{357}}{.08125/12}\right]} = \$743,970$$

The interest payment (which uses the PT rate) is

$$I = \left(\frac{.075}{12}\right)\$100M = \$625,000$$

and the scheduled principal payment (which uses the WAC rate) is

$$SP = \$743,970 - (.08125/12)(\$100M) = \$66,880.$$

The estimated prepaid principal using the 165% PSA model is:

$$CPR = 1.65\left(\frac{4}{30}\right).06 = .0132$$
$$SMM = 1 - [1 - .0132]^{1/12} = .0011067$$
$$PP = .0011067[\$100M - \$66,880] = \$110,600$$

The first-month cash flow (starting in month four) is:

$$CF_1 = \$625,000 + \$66,880 + \$110,600 = \$802,480$$

The beginning mortgage balance for month two is:

$$\$100M - \$66,880 - \$110,600 = \$99.822M$$

The scheduled monthly mortgage payment in the second month is:

$$p = \frac{\$99.822M}{\left[\frac{1-1/(1+(.08125/12))^{356}}{.08125/12}\right]} = \$743,140$$

The second-month interest payment is:

$$I = \left(\frac{.075}{12}\right)\$99.822M = \$623,890$$

The scheduled principal payment in month two is:

$$SP = \$743,140 - (.08125/12)(\$99.822M) = \$67,260$$

The estimated prepaid principal is:

$$CPR = 1.65\left(\frac{5}{30}\right).06 = .0165$$
$$SMM = 1 - [1 - .0165]^{1/12} = .00139$$
$$PP = .00189[\$99.822M - \$67,260] = \$138,210$$

Thus, the second-month cash flow is computed as:

$$CF_2 = \$623,890 + \$67,260 + \$138,210 = \$829,360$$

Table 3.1 shows the cash flows for the first few months.

The Excel spreadsheet MBS1.xls shows the complete computations for every month. Parameters can be changed (for different assumptions) to generate different cash flows.

Table 3.1

Month	Balance	Sch. Payment	Interest	Sch.Pr.	CPR	SMM	Prepayment	CF	Ending Bal.
1	100,000,000.00	743,967.06	625,000.00	66,883.73	0.0132	0.0011067	110,597.15	802,480.87	99,822,519.13
2	99,822,519.13	742,153.62	623,890.74	66,271.98	0.0165	0.0013855	138,213.22	828,375.94	99,618,033.93
3	99,618,033.93	740,145.25	622,612.71	65,648.15	0.0198	0.0016652	165,771.24	854,032.10	99,386,614.54
4	99,386,614.54	737,942.81	621,166.34	65,012.61	0.0231	0.0019457	193,248.74	879,427.69	99,128,353.20
5	99,128,353.20	735,547.31	619,552.21	64,365.76	0.0264	0.0022271	220,623.21	904,541.17	98,843,364.23
6	98,843,364.23	732,959.93	617,771.03	63,707.98	0.0297	0.0025093	247,872.18	929,351.19	98,531,784.07
7	98,531,784.07	730,181.99	615,823.65	63,039.70	0.0330	0.0027925	274,973.22	953,836.56	98,193,771.16
8	98,193,771.16	727,214.96	613,711.07	62,361.30	0.0363	0.0030765	301,903.97	977,976.35	97,829,505.88

Source: Johnson, S. (2004)

3.3 MBS Pricing and Quoting

The prices of an MBS are quoted as a percentage of the underlying mortgage balance. The mortgage balance at time t, F_t, is quoted as a proportion of the original balance. This is called the pool factor pf_t:

$$pf_t = \frac{F_t}{F_0} \tag{3.4}$$

Suppose, for example, an MBS backed by a collateral mortgage pool originally worth \$100M, a current pf of .92, and quoted at $95 - 16$ (note: 16 is 16/32) would have a market value of \$87.86M, as calculated:

$$\begin{aligned} F_t &= (pf_t)F_0 \\ &= (.92)(\$100M) = \$92M \end{aligned}$$

so that

$$\text{Market Value} = (.9550)(\$92M) = \$87.86M$$

The market value is the clean price; it does not take into account accrued interest, denoted AI. For an MBS, accrued interest is based on the time period from the settlement date (two days after the trade) to the first day of the next month. For example, if the time period is 20 days, the month is 30 days, and the WAC = 9%, then AI is \$.46M:

$$AI = \left(\frac{20}{30}\right)\left(\frac{.09}{12}\right)\$92M = \$460,000$$

The full market value would be \$88.32M:

$$\text{FullMktValue} = \$87.86M + \$460,000 = \$88.32M$$

The market price per share is the full market value divided by the number of shares. If the number of shares is 400, then the price of the MBS based on a $95 - 16$ quote would be \$220,080:

$$\textit{MBS price} = \frac{\$88.32M}{400} = \$220,800$$

The value of an MBS is equal to the present value (PV) of security's cash flows (CFs); thus, the value is a function of the MBS's expected CFs and the interest rate. In addition, for MBSs, the CFs are also dependent on rates R: A change in rates will change the prepayment of principal and either increase or decrease early CFs:

$$V_{MBS} = f(CFs, R)$$

where

$$CF = f(R).$$

Since cash flows, CFs, are a function of rates, the value of MBS is more sensitive to interest rate changes than a similar corporate bond. This sensitivity is known as *extension risk*. Note the following relationships:

$$\textit{if } R \downarrow \Rightarrow \text{lower discount rate} \Rightarrow V_M \uparrow \text{ (just like any other bond)}$$

and

$$\begin{aligned} \textit{if } R \downarrow &\Rightarrow \text{Increases prepayment} :\Rightarrow V_M \uparrow \\ &\Rightarrow \text{Earlier CFs} \uparrow \end{aligned}$$

On the other hand,

$$\textit{if } R \uparrow \Rightarrow \text{higher discount rate} \Rightarrow V_M \downarrow$$

and

$$\begin{aligned} \textit{if } R \uparrow &\Rightarrow \text{Decreases prepayment} :\Rightarrow V_M \downarrow \\ &\Rightarrow \text{Earlier CFs} \downarrow \end{aligned}$$

so that an increase in rates will reduce the market value of the MBS, leading to extension risk.

There are various exogenous and endogenous factors that influence prepayment other than refinancing rates. One is housing turnover—the long-term rate at which borrowers in a pool prepay their mortgages because they sell their homes. Another is the seasoning period, the number of months over which base voluntary prepayments (housing turnover, cash-out refinancing, and credit upgrades, but not rate refinancing or defaults) are assumed to increase to long-term levels. Other factors include credit curing—the long-term rate at which borrowers prepay their mortgages because improved credit and/or increased home pool prices enable them to get better rates and/or larger loans. As the pool burns out, the rate of curing declines.[2] Default, expressed as a percentage of the PSA Standard Default Assumption (SDA), affects prepayment, as well as the maximum rate-related CPR for burnout—CPR is lower for a pool that has experienced no prior rate-related refinancing. The lower the ratio, the faster the pool burns out.[3]

Many Wall Street firms use proprietary reduced-form prepayment models that use past prepayment rates and endogenous variables to explain prepayment. These models are calibrated to fit observed payment data, unrestricted by theoretical considerations.[4]

3.4 Prepayment Risk and Average Life of MBS

Average life is the weighted average of the MBS's or MBS collateral's time periods, with the weights being the periodic cash flow payments divided by the total principal. For example, the original average life of the 30-year, \$100M, 9%, 100 PSA mortgage (the first example in §3.2) portfolio is 12.077 years, computed as follows:

$$\textit{Ave. life} = \frac{1}{12}\frac{(1(\$71,295) + 2(\$88,376) + \ldots + 360(\$135,281))}{\$100,000,000} = 12.077$$

In general, the average life of the MBS can be computed by the following formula:

$$\textit{Ave. life} = \frac{1}{12}\frac{\sum_{i=1}^{360} i * CF_i}{F_0} \tag{3.5}$$

Prepayment risk can be measured in terms of how responsive (sensitive) an MBS's or MBS collateral's average life is to changes in prepayment speed (change in PSA) or equivalently to changes in rates (because rate changes are the major factor affecting speed):

$$\text{prepayment risk} = \frac{\Delta \textit{Ave. life}}{\Delta \textit{PSA}} \cong \frac{\Delta \textit{Ave. life}}{\Delta R} \tag{3.6}$$

An MBS or its collateral would have zero prepayment risk if

$$\text{prepayment risk} = \frac{\Delta \textit{Ave. life}}{\Delta \textit{PSA}} = 0.$$

One of the more significant innovations in finance occurred in the 1980s with the development of derivative MBSs, such as Planned Amortization Classes (PACs), which had different prepayment risk features, including some derivatives with zero prepayment risk.

Assumptions of prepayment rates can be made based on the probability of refinancing rates changing. For example, if there is a high probability that the Federal Open Markets Committee (the Fed) will lower rates (based, for example, on media reports that they intend to do so in the near future), refinancing rates can be expected to fall as well so that more homeowners will refinance their mortgages at lower rates. This in turn will increase the speed of prepayment and thus of the cash flows to investors. PSA rates should then be adjusted upward. Conversely, if the Fed is expected to raise rates as a response to, say, inflation, refinancing rates can be expected to rise, decreasing prepayment risk and lengthening the average life of the MBS. PSA rates should then be adjusted downward.

The best way to model refinancing rate scenarios is through Monte Carlo simulation. One first constructs an interest rate tree—i.e., a binomial tree[5]—with both the spot rates and refinancing rates at each node. One runs many simulation paths sampling from possible interest rate paths that rates could possibly take in the tree. For each simulation path, one estimates the cash flows based on the refinancing rates at each step along the path. (Each time step along the path corresponds to a time step made in the short-rate tree.) Specifically, Monte Carlo simulation can be used to determine the MBS's theoretical value or rate of return through the following steps:

1. Simulate interest rates. Use a binomial interest-rate tree to generate different paths for spot rates and refinancing rates.

2. Specify a prepayment model based on the spot rates.

3. Generate CF paths for a mortgage portfolio, MBS, or tranche.

4. Determine the PV of each path, the distribution of the path, the average (theoretical value), and standard deviation. Alternatively, given the market value, determine each path's rate of return, distribution, average, and standard deviation.

Step 1

In step 1, to simulate interest rates, we generate interest rate paths from a binomial interest-rate tree.[6] For example, assume a three-period binomial tree of one-year spot rates, R_t^S, and refinancing rates, R_t^{ref}, where $R_0^S = 6\%$, $R_0^{ref} = 8\%$, $u = 1.1$, and $d = .9091 = 1/1.1$. With three periods, there are four possible rates after three periods (years), and there are eight possible paths in the binomial tree shown in Figure 3.3. Table 3.2 shows eight short-rate paths simulated from the preceding binomial tree (the eight possible paths rates can take in the tree).

Suppose we have a mortgage portfolio with a par value of \$1M, a WAM $= 10$ years, a WAC $= 8\%$, PT rate $= 8\%$, annual cash flow payments, the mortgages are insured against default risk, and has a balloon payment at the end of year 4 equal to the balance at the

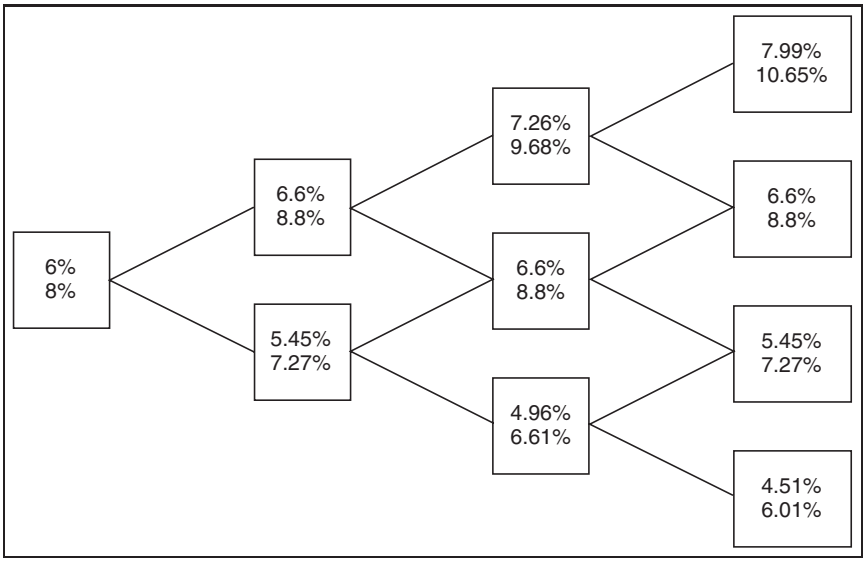

Figure 3.3 Binomial tree for spot and refinancing rates.

Table 3.2

	Year 1	Year 2	Year 3	Year 4
Path 1	8.0000%	7.2728%	6.6117%	6.0107%
Path 2	8.0000%	7.2728%	6.6117%	7.2728%
Path 3	8.0000%	7.2728%	8.0000%	7.2728%
Path 4	8.0000%	8.8000%	8.0000%	7.2728%
Path 5	8.0000%	7.2728%	8.0000%	8.0000%
Path 6	8.0000%	8.8000%	8.0000%	8.0000%
Path 7	8.0000%	8.8000%	9.6800%	8.8000%
Path 8	8.0000%	8.8000%	9.6800%	10.6480%

beginning of year 4 (e.g. the scheduled principal in year 4). We compute the scheduled monthly mortgage payment:

$$p = \frac{\$1,000,000}{\frac{1-(1/1.08)^{10}}{.08}} = \$149,029$$

If we initially assume no prepayment risk, then we obtain the cash flows shown in Table 3.3. The balloon payment at the end of year 4 is:

$$\begin{aligned} \textit{Balloon} &= \textit{Balance}(\textit{yr4}) - \textit{Sch.prin}(\textit{yr4}) \\ &= \$775,149 - \$86,957 = \$688,946 \end{aligned}$$

The cash flow in year 4 can be computed as

$$\begin{aligned} CF_4 &= \textit{Balloon} + p \\ &= \$688,946 + \$149,029 = \$837,973 \end{aligned}$$

or equivalently, as

$$\begin{aligned} CF_4 &= \textit{Balance}(\textit{yr4}) + \textit{Interest} \\ &= \$775,903 + \$62,513 = \$837,973 \end{aligned}$$

Table 3.3

Year	Balance	P	Interest	Scheduled Principal	CF
1	$1,000,000	$149,029	$80,000	$69,029	$149,029
2	$930,970	$149,029	$74,478	$74,551	$149,029
3	$856,419	$149,029	$68,513	$80,516	$149,029
4	$775,903	$149,029	$62,072	$86,957	$837,975

Step 2

The second step of the Monte Carlo process is to specify a prepayment model. Suppose we specify the prepayment schedule shown in Table 3.4. The CPR is determined by the value of the spread $X = WAC - R^{ref}$.

Table 3.4

Range					CPR
		X	$\leq$	0	5%
0	$<$	X	$\leq$	0.5%	10%
0.5%	$<$	X	$\leq$	1.00%	20%
1.00%	$<$	X	$\leq$	1.25%	30%
1.25%	$<$	X	$\leq$	2.0%	40%
2.0%	$<$	X	$\leq$	2.5%	50%
2.5%	$<$	X	$\leq$	3.0%	60%
		X	$>$	3.0%	70%

Step 3

The third step of the valuation process is the estimation of cash flows for each path based on the simulated path of refinancing rates, the spread X, and thus the CPR (see Table 3.5).

The calculation of the cash flows for the first path are shown as follows. In year 1, the scheduled mortgage payment is:

$$p = \frac{\$1,000,000}{\frac{1-(1/1.08)^{10}}{.08}} = \$149,029$$

The interest payment is:

$$I = 0.08(\$1,000,000) = \$80,000$$

The scheduled principal is:

$$SP = \$149,029 - \$80,000 = \$69,029$$

Table 3.5 (continued next page)

Path 1 Year	1 Ref	2 Balance	WAC	3 Interest	4 Sch. Prin.	5 CPR	6 Prepaid	7 CF
1	0.072728	1000000	0.08	80000	69029.49	0.2	186194.1	335223.6
2	0.066117	744776.4	0.08	59582.11	59641.48	0.4	274054	393277.6
3	0.060107	411081	0.08	32886.48	38647.68	0.4	148973.3	220507.5
4		223460	0.08	17876.8				241336.8

Path 2 Year	1 Ref	2 Balance	WAC	3 Interest	4 Sch. Prin.	5 CPR	6 Prepaid	7 CF
1	0.072728	1000000	0.08	80000	69029.49	0.2	186194.1	335223.6
2	0.066117	744776.4	0.08	59582.11	59641.48	0.4	274054	393277.6
3	0.072728	411081	0.08	32886.48	38647.68	0.2	74486.66	146020.8
4		297946.6	0.08	23835.73				321782.4

Source: Johnson, S. (2004)

Table 3.5 (continued)

Path 3	1	2		3	4	5	6	7
Year	Ref	Balance	WAC	Interest	Sch. Prin.	CPR	Prepaid	CF
1	0.072728	1000000	0.08	80000	69029.49	0.2	186194.1	335223.6
2	0.08	744776.4	0.08	59582.11	59641.48	0.05	34256.75	153480.3
3	0.072728	650878.2	0.08	52070.25	61192.16	0.2	117937.2	231199.6
4		471748.8	0.08	37739.91				509488.7

Path 4	1	2		3	4	5	6	7
Year	Ref	Balance	WAC	Interest	Sch. Prin.	CPR	Prepaid	CF
1	0.088	1000000	0.08	80000	69029.49	0.05	46548.53	195578
2	0.08	884422	0.08	70753.76	70824.26	0.05	40679.89	182257.9
3	0.072728	772917.8	0.08	61833.43	72665.69	0.2	140050.4	274549.5
4		560201.7	0.08	44816.14				605017.9

Path 5	1	2		3	4	5	6	7
Year	Ref	Balance	WAC	Interest	Sch. Prin.	CPR	Prepaid	CF
1	0.072728	1000000	0.08	80000	69029.49	0.2	186194.1	335223.6
2	0.08	744776.4	0.08	59582.11	59641.48	0.05	34256.75	153480.3
3	0.088	650878.2	0.08	52070.25	61192.16	0.05	29484.3	142746.7
4		560201.7	0.08	44816.14				605017.9

Path 6	1	2		3	4	5	6	7
Year	Ref	Balance	WAC	Interest	Sch. Prin.	CPR	Prepaid	CF
1	0.088	1000000	0.08	80000	69029.49	0.05	46548.53	195578
2	0.08	884422	0.08	70753.76	70824.26	0.05	40679.89	182257.9
3	0.088	772917.8	0.08	61833.43	72665.69	0.05	35012.61	169511.7
4		665239.5	0.08	53219.16				718458.7

Path 7	1	2		3	4	5	6	7
Year	Ref	Balance	WAC	Interest	Sch. Prin.	CPR	Prepaid	CF
1	0.088	1000000	0.08	80000	69029.49	0.05	46548.53	195578
2	0.096	884422	0.08	70753.76	70824.26	0.05	40679.89	182257.9
3	0.088	772917.8	0.08	61833.43	72665.69	0.05	35012.61	169511.7
4		665239.5	0.08	53219.16				718458.7

Path 8	1	2		3	4	5	6	7
Year	Ref	Balance	WAC	Interest	Sch. Prin.	CPR	Prepaid	CF
1	0.088	1000000	0.08	80000	69029.49	0.05	46548.53	195578
2	0.096	884422	0.08	70753.76	70824.26	0.05	40679.89	182257.9
3	0.10648	772917.8	0.08	61833.43	72665.69	0.05	35012.61	169511.7
4		665239.5	0.08	53219.16				718458.7

Source: Johnson, S. (2004)

The prepaid principal is:

$$PP = 0.20(\$1,000,000 - \$69,029) = \$186,194$$

The cash flow in year 1 is:

$$CF_1 = \$80,000 + \$69,029 + \$186,194 = \$335,223$$

For year 2, along path 1, we have a balance of:

$$B_2 = \$1,000,000 - \$69,029 - \$186,194 = \$744,776$$

The scheduled monthly mortgage payment is:

$$p = \frac{\$744,446}{\frac{1-(1/1.08)^9}{.08}} = \$119,223$$

The interest payment is:

$$I = 0.08(\$744,776) = \$59,582$$

The scheduled principal payment is:

$$SP = \$119,223 - \$59,582 = \$59,641$$

The prepaid principal in the second year is:

$$PP = 0.4(\$755,776 - \$59,641) = \$274,054$$

The cash flow is:

$$CF_2 = \$59,582 + \$59,641 + \$274,052 = \$393,277$$

In year 3, on path 1, the balance is:

$$B_3 = \$744,776 - \$59,641 - \$274,054 = \$411,081$$

The scheduled monthly mortgage payment is:

$$p = \frac{\$411,081}{\frac{1-(1/1.08)^8}{.08}} = \$71,543$$

The interest payment is:

$$I = 0.08(\$411,081) = \$32,886$$

The scheduled principal payment is:

$$SP = \$71,543 - \$32,886 = \$38,648$$

The prepaid principal in the third year is:

$$PP = 0.4(\$411,081 - \$38,648) = \$148,973$$

The cash flow is:

$$CF_3 = \$32,886 + \$38,648 + \$148,973 = \$220,507$$

Finally, in year 4, the balance is:

$$B_4 = \$411,081 - \$38,648 - \$148,943 = \$223,460$$

The interest payment is:

$$I = 0.08(\$223,460) = \$17,877$$

The cash flow is:

$$\begin{aligned} CF_4 &= B_4 + I \\ &= \$223,460 + \$17,877 = \$241,337 \end{aligned}$$

The cash flows for all the other paths are computed similarly.

Step 4

The fourth step of the valuation process is the valuation of the cash flows along each of the paths. The PV of each path's cash flows are determined by specifying the appropriate discount rates. Because the mortgages are insured against default risk, the only risk investors are exposed to is prepayment risk. The risk premium for such risk is known as the *option adjusted spread* (OAS). The OAS is a measure of the spread over the government Treasury bonds rates provided by the MBS when all embedded options have been into account.[7] One can view the OAS as the market price for unmodeled risks (risks that the model cannot capture), such as the forecast error associated with prepayments. The OAS is the spread, such that when added to all the spot rates on all interest rate paths, make the average present value of the paths equal to the observed market price (plus accrued interest). Thus, it equates the observed market *price* of a security to its theoretical *value*. Mathematically, it is equivalent to the solution of K in

$$\begin{aligned} P^{Market} &= \frac{1}{N}\left[PV(path\ 1) + PV(path\ 2) + \ldots + PV(path\ N)\right] \\ &= \frac{1}{N}\left[\sum_{i=1}^{T}\frac{CF_i^{path\ 1}}{(1+Z_i^1+K)^i} + \sum_{i=1}^{T}\frac{CF_i^{path\ 2}}{(1+Z_i^2+K)^i} + \ldots + \right. \\ &\qquad \left. \sum_{i=1}^{T}\frac{CF_i^{path\ N}}{(1+Z_i^N+K)^i}\right] \end{aligned} \tag{3.7}$$

where Z_i^j is the zero rate at time i—i.e, month $i = 1, \ldots, T$ on path $j = 1, ..., N$. Typically, $T = 360$ and $N = 1,024$.

The cash flow "yield" that is a standard measure in evaluating any MBS is the *static spread*. This is the yield spread in "a static scenario (i.e., no volatility interest rates) of the bond over the theoretical Treasury spot rate curve, not a single point on the Treasury yield curve."[8] The magnitude of this spread depends on the steepness of the yields curve: the steeper the curve, the greater the difference between the bond and Treasury yields.[9] There are two ways to compute the static spread. The first approach is to use today's yield curve to discount future cash flows and keep the mortgage refinancing rate fixed at today's mortgage rate.[10]

Because the mortgage refinancing rate is fixed, the investor can usually specify a reasonable prepayment rate, which can be used to estimate the bond's future cash flows until the maturity of the bond. The second approach, known as the *zero volatility OAS*, computes the static spread by allowing the mortgage rates to go up the curve as implied by forward interest rates.[11] In this case, a prepayment model is needed to determine the vector of future prepayment rates (a prepayment schedule) implied by the vector of future refinancing rates. After a static spread and OAS is computed, the implied cost of the prepayment option embedded in any MBS can be computed by calculating the different between the OAS (at the assumed volatility of interest rates) and the static spread. That is

$$\textit{Option cost} = \textit{Static spread} - \textit{OAS} \tag{3.8}$$

Consequently, because, in general, a tranche's option cost is more stable than its OAS in the face of uncertainty of interest rate movements, then, for small market moves, the OAS of a tranche may be approximated by recalculating the static spread and subtracting its option cost. This is quite useful because the OAS is computationally expensive to evaluate while the static spread is cheap and easy to compute.[12]

It is important to point out that investors in MBSs hold the equivalent of long positions in noncallable bonds and short positions in call (prepayment) options.[13] The noncallable bond is a collection of zero-coupon bonds—i.e., Treasury strips—and the call option gives the borrower the right to prepay the mortgage at any time prior to maturity of the loan.[14] Thus, the value of MBSs is the difference between the value of the noncallable bond and the value of the call (prepayment) option. The OAS is the spread differential between the bond component and the option value component of the MBS. The two main inputs into the computation of an OAS are the cash flows generated as a function of the principal (scheduled and unscheduled) and coupon payments, as well as the interest rate paths generated under an assumed term structure of the zero-coupon curve for discounting the cash flows.[15] At each cash flow date, the spot rate (observed from the interest rate path taken at the corresponding time step of the term structure) determines the discount factor for each cash flow.[16]

Denote z_t to be the appropriate zero discount rate for maturity t (i.e., t years or months), seen today (time 0) (and similarly, f_{tj}, the forward discount rate of maturity t seen at time j), and K, the option adjusted spread. In our simple four-step binomial example, the one-year forward (zero) rate at time 0 is $f_{10} = 8.0\%$; the one-year forward rates at time step 1 are $f_{11} = \{8.6\%, 7.45\%\}$; the one-year forward rates at time step 2 are $f_{12} = \{9.26\%, 8\%, 6.96\%\}$; and the one-year forward rates at time step 3 are $f_{13} = \{9.986\%, 8.6\%, 7.45\%, 6.51\%\}$.

The value of each path is obtained by discounting each cash flow by its risk-adjusted zero-spot rate, z. In our example of four time steps, the value of the MBS on path i is

$$V_i = \frac{CF_1}{1+z_1} + \frac{CF_2}{(1+z_2)^2} + \frac{CF_3}{(1+z_3)^3} + \frac{CF_4}{(1+z_4)^4}$$

where, because we can express zero rates in terms of forward rates, we have

$$\begin{aligned}
z_1 &= f_{10}, \\
z_2 &= ((1+f_{10})(1+f_{11}))^{1/2} - 1 \\
z_3 &= ((1+f_{10})(1+f_{11})(1+f_{12}))^{1/3} - 1 \\
z_4 &= ((1+f_{10})(1+f_{11})(1+f_{12})(1+f_{13}))^{1/4} - 1
\end{aligned}$$

In general, on path $i = 1, \ldots, N$,

$$z_T = \{(1+f_{10})(1+f_{11})(1+f_{12})...(1+f_{12T})\}^{1/T} - 1$$

Note that in our example, we have assumed one-year forward rates, but in a more complex and realistic implementation, we would be simulating future one-month rates over a period of 360 months. Thus, for each path, we would be simulating 360 one-month future interest rates, mortgage refinancing rates, and cash flows instead of just four, and we would be simulating many more paths—i.e., 1024, instead of eight.

The zero-rate calculations for path one are

$$\begin{aligned}
z_1 &= 0.08 \\
z_2 &= ((1.08)(1.074546))^{1/2} - 1 = 0.077269 \\
z_3 &= ((1.08)(1.074546)(1.069588))^{1/3} - 1 = 0.074703 \\
z_4 &= ((1.08)(1.074546)(1.069588)(1.06508))^{1/4} - 1 = 0.072289
\end{aligned}$$

so that the MBS value for path one is

$$V_1 = \frac{\$335,224}{1.08} + \frac{\$393,278}{(1.077269)^2} + \frac{\$220,507}{(1.04703)^3} + \frac{\$241,337}{(1.072289)^4} = \$1,009,470.$$

Table 3.6 shows the MBS computed for each of the eight paths.

The final step is to compute the theoretical value of the MBS by averaging over all values taken on each path. In this example, the theoretical value of the mortgage portfolio is the average of the MBS values computed on each of the eight paths:

$$\bar{V} = \frac{1}{N}\sum_{i=1}^{N} V_i \tag{3.9}$$

Evaluating (3.8), the theoretical value is \$997,235 or 99.7235% of par. Note, in addition to the theoretical value, we also can determine the variance of the distribution:

$$Var(V) = \frac{1}{N}\sum_{i=1}^{N}[V_i - \bar{V}]^2 \tag{3.10}$$

Equivalently, we can also compute the theoretical value by taking the weighted average of each MBS value computed on each path, where the weight is the probability of obtaining that value on the path (each up and down move is assumed to be 0.5), shown in Table 3.7.

Table 3.6 (continued next page)

Path 1 Year	7 CF	8 Z1,t-1	9 Zt0	10 Value	11 Prob.
1	335223.6	0.08	0.08	310392.2	0.5
2	393277.6	0.074546	0.07727	338883.5	0.5
3	220507.5	0.069588	0.074703	177647.1	0.5
4	241336.8	0.06508	0.072289	182547.5	
			Value =	1009470	0.125

Path 2 Year	7 CF	8 Z1,t-1	9 Zt0	10 Value	11 Prob.
1	335223.6	0.08	0.08	310392.2	0.5
2	393277.6	0.074546	0.07727	338883.5	0.5
3	146020.8	0.069588	0.074703	117638.5	0.5
4	321782.4	0.074546	0.074664	241252.6	
			Value =	1008167	0.125

Source: Johnson, S. (2004)

Table 3.6 (continued)

Path 3 Year	7 CF	8 Z1,t-1	9 Zt0	10 Value	11 Prob.
1	335223.6	0.08	0.08	310392.2	0.5
2	153480.3	0.074546	0.07727	132252.5	0.5
3	231199.6	0.08	0.078179	184465.3	0.5
4	509488.7	0.074546	0.07727	378300.6	
			Value =	1005411	0.125

Path 4 Year	7 CF	8 Z1,t-1	9 Zt0	10 Value	11 Prob.
1	195578	0.08	0.08	181090.8	0.5
2	182257.9	0.086	0.082996	155393.5	0.5
3	274549.5	0.08	0.081996	216742.2	0.5
4	605017.9	0.074546	0.080129	444493.9	
			Value =	997720.3	0.125

Path 5 Year	7 CF	8 Z1,t-1	9 Zt0	10 Value	11 Prob.
1	335223.6	0.08	0.08	310392.2	0.5
2	153480.3	0.074546	0.07727	132252.5	0.5
3	142746.7	0.08	0.078179	113892.1	0.5
4	605017.9	0.086	0.080129	444493.9	
			Value =	1001031	0.125

Path 6 Year	7 CF	8 Z1,t-1	9 Zt0	10 Value	11 Prob.
1	195578	0.08	0.08	181090.8	0.5
2	182257.9	0.086	0.082996	155393.5	0.5
3	169511.7	0.08	0.081996	133820.4	0.5
4	718458.7	0.086	0.082996	522269.5	
			Value =	992574 1	0.125

Path 7 Year	7 CF	8 Z1,t-1	9 Zt0	10 Value	11 Prob.
1	195578	0.08	0.08	181090.8	0.5
2	182257.9	0.086	0.082996	155393.5	0.5
3	169511.7	0.0926	0.086188	132277.2	0.5
4	718458.7	0.086	0.086141	516246.6	
			Value =	985008	0.125

Path 8 Year	7 CF	8 Z1,t-1	9 Zt0	10 Value	11 Prob.
1	195578	0.08	0.08	181090.8	0.5
2	182257.9	0.086	0.082996	155393.5	0.5
3	169511.7	0.0926	0.086188	132277.2	0.5
4	718458.7	0.09986	0.08959	509741.1	
			Value =	978502.5	0.125

Source: Johnson, S. (2004)

Table 3.7

Value	Prob.
1009470	0.125
1008167	0.125
1005411	0.125
997720.3	0.125
1001031	0.125
992574.1	0.125
985008	0.125
978502.5	0.125
Wt. Value	**$997,235**

3.5 MBS Pricing Using Monte Carlo in C++

To price MBSs in C++, we create and define an MBS class that contains methods for MBS pricing via Monte Carlo simulations of spot rate paths in a binomial tree.

```
#ifndef _MBS_H__
#define _MBS_H__

#include <vector>
#include "math.h"
#include "time.h"
#include "Utility.h"
#include "TNT.h"
#define SIZE_X 100
#define SIZE_Y 100

using namespace std;
```

We define two global double array variables that will be used to store the spot rates and discount rates in the binomial tree.

```
static TNT::Array2D<double> spotRate(SIZE_X,SIZE_Y);
static TNT::Array2D<double> discountRate(SIZE_X,SIZE_Y);
```

The *MBS* class contains an overloaded constructor that accepts the notional principal, coupon, weighted average WAC, weighted average maturity (WAM), and option adjusted spread (OAS). The class contains a method `calcPrice` that first builds a binomial tree and then simulates the interest rate paths on the tree using Monte Carlo. The `calcPrice` method accepts the initial spot rate, mortgage refinance rate, the number of steps in the binomial tree, and the number of simulations. The *MBS* class also contains a method to compute the conditional prepayment rate (CPR) `calcCPR`, which accepts the current refinance rate and a method `computeZeroRates` that computes the current discount factor by accepting as input the current time step in the binomial tree and the stored history of discount rates on the current path. The MBS class contains a `calcPayment` function that computes the current mortgage payment by receiving the remaining principal and time to maturity as

input. Finally, the MBS class contains a `getPrice` that returns the calculated MBS price, a `getStdDev` method that returns the standard deviation of the computed MBS price, and a `getStdError` method that returns the standard error of the computed MBS price.

MBS.h

```
// MBS.h: interface for the MBS class.
//
//////////////////////////////////////////////////////////////////////

#ifndef _MBS_H__
#define _MBS_H__

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
#include <vector>
#include "math.h"
#include "time.h"
#include "Utility.h"
#include "TNT\TNT.h"
#define SIZE_X 100
#define SIZE_Y 100

using namespace std;
static TNT::Array2D<double> spotRate(SIZE_X,SIZE_Y);
static TNT::Array2D<double> discountRate(SIZE_X,SIZE_Y);

class MBS
{
  public:
        MBS();
        MBS(double principal, double coupon, double WAC, double WAM,
          double OAS) :
                faceValue(principal), coupon(coupon), WAC(WAC), WAM(WAM),
                OAS(OAS), T(WAM) { }
        virtual ~MBS()  { }
        double calcPayment(double principal, double T);   // compute
                                                          // payment
                                                          // amount
        void calcPrice(double initRate, double financeRate, int N,
          long int M);
        double calcCPR(double rate);
        void buildTree(double initRate, double financeRate, int N);
        double computeZeroRates(int cnt, vector<double> rate);
        double calcSMM(double x);
        double getPrice();
        double getStdDev();
        double getStdErr();
        double getMaturity();
        double getWAM();
        double getWAC();
        double getOAS();
  private:
        double OAS;               // option adjusted spread
        double faceValue;         // principal amount
```

```
        double coupon;            // coupon rate
        double WAM;               // weighted average maturity
        double WAC;               // weighted average coupon
        vector<double> zeroRates; // store discount zero coupon rates
        double T;                 // maturity of MBS
        double mbsPrice;          // price
        double stdDev;            // standard deviation
        double stdErr;            // standard error
};

#endif _MBS_H__
```

The method definitions are

MBS.cpp

```
// MBS.cpp: implementation of the MBS class.
//
//////////////////////////////////////////////////////////////////////

#include "MBS.h"

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

void MBS::buildTree(double initRate, double financeRate, int N)
{

        Utility util;
        double u = 1.1;
        double d = 1/u;
        double p = (exp(initRate*T) - d)/(u - d);
        double deviate = 0.0;
        long seed = 0;
        double refRate = financeRate;
        long* idum = 0;
        double pay = faceValue;
        double faceAmount = 0.0;
        double interest = 0.0;
        double schedulePrincipal = 0.0;
        double prepaidPrincipal = 0.0;
        double CPR = 0.0;
        double balance = faceValue;
        double sum = 0.0;
        double totalsum = 0.0;
        double SMM = 0.0;
        TNT::Array1D<double> CF(SIZE_X);   // cash_flow
        vector<double> disc(0.0);

        srand(unsigned(time(0)));
        seed = (long) rand() % 100;
        idum = &seed;
        // build binomial tree for rates
        for (int i = 0; i <= N; i++)
        {
```

```
        for (int j = 0; j <= i; j++)
        {
                spotRate[i][j] = initRate*pow(u,j)*pow(d,i-j);
                discountRate[i][j] = spotRate[i][j] + OAS;
        }
}


faceAmount = faceValue;
int k = 0;
long int M = 10000;
int cnt = 0;
double r = 0.0;
int j = 0;

for (k = 0; k < M; k++)
{
        sum = 0.0;
        balance = faceValue;
        refRate = financeRate;
        j = 0;
        disc.clear();
        disc.empty();
        disc.push_back(discountRate[0][0]);

        for (i = 0; i < N; i++)
        {
                balance = balance - (schedulePrincipal +
                            prepaidPrincipal);
                deviate = util.gasdev(idum);

                if (deviate > 0)
                {
                        j++;
                        refRate = refRate*u;
                }
                else
                {
                        j--;
                        if (j < 0)
                                j = 0;
                        refRate = refRate*d;
                }
                disc.push_back(discountRate[i+1][j]);
                interest = coupon*balance;
                pay = calcPayment(balance,WAM-i);
                schedulePrincipal = pay - interest;

                if (balance >= schedulePrincipal)
                {
                        CPR = calcCPR(refRate);
                        SMM = calcSMM(CPR);
                        prepaidPrincipal = SMM*(balance -
                          schedulePrincipal);

                        if (i != N-1)
                                CF[i] = interest +
```

```
                                                schedulePrincipal +
                                                prepaidPrincipal;
                                else
                                        CF[i] = interest + balance;

                                r = computeZeroRates(i,disc);
                                sum = sum + CF[i]/(pow(1+r,i+1));

                        }
                        else
                                goto x;

                }
                x:
                totalsum = totalsum + sum;
        }
        double ave = (totalsum/M);
        std::cout << "MBS price = " << ave << endl;
}

double MBS::calcCPR(double rate)
{
        double CPR = 0.0;
        double value = WAC - rate;

        /*
        if (value <= 0)
                CPR = 0.05;
        else if ((value <= 0.005) && (value > 0))
                CPR = 0.10;
        else if ((value <= 0.01) && (value > 0.005))
                CPR = 0.20;
        else if ((value <= 0.0125) && (value > 0.01))
                CPR = 0.30;
        else if ((value <= 0.02) && (value > 0.0125))
                CPR = 0.40;
        else if ((value <= 0.025) && (value > 0.02))
                CPR = 0.50;
        else if ((value <= 0.03) && (value > 0.025))
                CPR = 0.60;
        else
                CPR = 0.70;
        */

        CPR = 100*(1-pow((1-(value/100)),12));

        return CPR;

}

double MBS::calcPayment(double fv, double T) {
        return (fv*coupon)/(1-pow(1/(1+coupon),T));
}

void MBS::calcPrice(double initRate, double financeRate, int N,
  long int M){
```

```
Utility util;      // utility class for generating
                   // random deviates
double u = 1.1;    // up move in binomial tree
double d = 1/u;    // down move in binomial tree
double p = (exp(initRate*T) - d)/(u - d);  // up probablity
double deviate = 0.0;                      // random deviate
long seed = 0;                             // seed
double refRate = financeRate;    // refinance rate
long* idum = NULL;               // pointer to seed value for RNG
double pay = faceValue;          // face value of MBS
double faceAmount = 0.0;         // face amount
double interest = 0.0;           // interest payment
double schedulePrincipal = 0.0;  // scheduled principal payments
double prepaidPrincipal = 0.0;   // prepaid principal payments
double CPR = 0.0;                // conditional prepayments
double SMM = 0.0;                // monthly mortality
double balance = faceValue;      // balance remaining
double sum = 0.0;                // sum of discounted cash flows
                                 // along a path
double totalsum = 0.0;  // total sum of all discounted cash flows
double totalsum2 = 0.0;
TNT::Array1D<double> CF(SIZE_X); // cash_flow
vector<double> disc(0.0);        // stores discount rates

// build binomial tree for rates
for (int i = 0; i <= N; i++)
{
        for (int j = 0; j <= i; j++)
        {
                spotRate[i][j] = initRate*pow(u,j)*pow(d,i-j);
                discountRate[i][j] = spotRate[i][j] + OAS;
        }
}

srand(unsigned(time(0)));
seed = (long) rand() % 100;
idum = &seed;
faceAmount = faceValue;
int k = 0;
int cnt = 0;
double r = 0.0;
int j = 0;

for (k = 0; k < M; k++)
{
        sum = 0.0;
        balance = faceValue;
        refRate = financeRate;
        j = 0;
        disc.clear();
        disc.push_back(discountRate[0][0]);

        for (i = 0; i < N; i++)
        {
                balance = balance - (schedulePrincipal +
                            prepaidPrincipal);
                deviate = util.gasdev(idum);
```

```
                        if (deviate > 0)
                        {
                               j++;
                               refRate = refRate*u;
                        }
                        else
                        {
                               j--;
                               if (j < 0)
                                j = 0;
                               refRate = refRate*d;
                        }
                        disc.push_back(discountRate[i+1][j]);
                        interest = coupon*balance;
                        pay = calcPayment(balance,WAM-i);
                        schedulePrincipal = pay - interest;

                        if (balance >= schedulePrincipal)
                        {
                           CPR = calcCPR(refRate);
                           SMM = calcSMM(CPR);
                             prepaidPrincipal = SMM*(balance -
                                                schedulePrincipal);

                             if (i != N-1)
                                CF[i] = interest + schedulePrincipal +
                                            prepaidPrincipal;
                             else
                                CF[i] = interest + balance;

                            r = computeZeroRates(i,disc);
                            sum = sum + CF[i]/(pow(1+r,i+1));

                        }
                        else   // break out of loop
                           goto x;

                }
                x:
                  totalsum = totalsum + sum;
                  totalsum2 = totalsum2 + sum*sum;
        }
        double ave = (totalsum/M);

        mbsPrice = ave;
        stdDev = sqrt(totalsum2 - (double)(totalsum*totalsum)/M)*(exp(-
        2*initRate*T)/(M-1));
        stdErr = (double) stdDev/sqrt(M);
}


double MBS::calcSMM(double CPR) {
         return (1 - pow((1 - CPR),(double)1/12));
}

double MBS::computeZeroRates(int cnt, vector<double> rate) {
```

```
        double value = WAC+1;
        for (int j = 1; j <= cnt; j++)
            value = value*(1 + rate[j]);

        if (cnt == 0)
            value = WAC;
        else
           value = pow(value,(double)1/(cnt+1)) - 1;

        return value;
}

double MBS::getPrice() {
     return mbsPrice;
}

double MBS::getStdDev() {
     return stdDev;
}

double MBS::getStdErr() {
     return stdErr;
}

double MBS::getMaturity() {
        return T;
}

double MBS::getWAM() {
        return WAM;
}

double MBS::getWAC() {
        return WAC;
}

double MBS::getOAS() {
        return OAS;
}
```

Consider pricing an MBS with the parameters used previously:

Main.cpp

```
#include <fstream.h>
#include <stdlib.h>
#include <iostream.h>
#include <string.h>
#include <math.h>
#include <map>

#define SIZE_X 100
#include "CMO.h"

void main()
```

```
{
        std::cout.precision(7);
        double principal = 1000000;      //  underlying principal
                                         //  (notional) of MBS
        double coupon = 0.08;            //  coupon rate
        double WAC = 0.08;               //  weighted average
                                         //  coupon rate
        double WAM = 10;                 //  weighted average maturity
        double OAS = 0.02;               //  option adjusted spread
        double initSpotRate = 0.06;      //  spot rate
        double initRefinanceRate = 0.08; //  refinance rate
        int N = 10;                      //  number of time steps in tree
        long int M = 100000;             //  number of simulation paths
        MBS mbs(principal,coupon,WAC,WAM,OAS);

        std::cout << "Running Monte Carlo to price MBS..." << endl << endl;
        mbs.calcPrice(initSpotRate,initRefinanceRate,N,M);
        std::cout << "MBS Price = " << mbs.getPrice() << endl;
        std::cout << "Std Deviation = " << mbs.getStdDev() << endl;
        std::cout << "Std Error = " << mbs.getStdErr() << endl << endl;

        std::cout << "Pricing MBS with Simulations of Binomial
          Tree Paths..." << endl;
        MBS mbs1(principal,coupon,WAC,N,OAS);
        mbs1.buildTree(initSpotRate,coupon,N);

        vector<Tranche> tranche;
        Tranche trA('A',500000,0.06);
        tranche.push_back(trA);
        Tranche trB('B',300000,0.065);
        tranche.push_back(trB);
        Tranche trC('C',200000,0.07);
        tranche.push_back(trC);
        Tranche trZ('Z',100000, 0.075);
        tranche.push_back(trZ);

        std::cout << endl;
        std::cout << "Pricing CMO Tranches..." << endl << endl;
        CMO cmo(mbs,tranche);
        cmo.calcCashFlows(initSpotRate,initRefinanceRate,N,M);
}
```

The results are as follows:

```
MBS Price = 964386.69
Std Deviation = 110.07
Std Error = 1.10
```

We can improve the accuracy by increasing the number of simulations. For instance, if M $= 100,000$, then:

```
MBS Price = 964469.78
Std Deviation = 34.86
Std Error = 0.11
```

Thus, the price of the MBS is priced at roughly 96.5% of par. The more time steps, however, improves the accuracy of the computed price.

Continuous Time Model

The binomial model is a simple discrete model and does not capture the movement of interest rates in practice because at each step, rates can only go up or down—they cannot stay the same or move in between time steps. To capture a realistic evolution of interest rate movements, an arbitrage-free model of the term structure of interest rates is typically used. The short rate is assumed to follow a diffusion (a continuous time stochastic) process. The general form of these models is described in terms of changes in the short rate, as follows:

$$dr_t = \kappa(\theta - r)dt + \sigma r^{\alpha} dz_t, \quad r(0) = r_0$$

where dr_t represents an infinitesimal change in r_t over an infinitesimal time period, dt, and dz_t is a standard Wiener process. κ is the speed of mean-reversion, θ is the long-run mean of the interest rate process, α is the proportion conditional volatility exponent, and σ is the instantaneous standard deviation of changes in r_t. The various short-rate models differ by the parameter α. The Vasicek model assumes it is 0, the Cox-Ingersoll-Ross (CIR) model assumes it to be 0.5, and the Courtadon model assumes it to be 1.

In order to simulate the process, we discretize it as follows (assume Courtadon):

$$\Delta r_t = \kappa(\theta - r_t)\Delta t + \sigma r_t \sqrt{\Delta t} z_t, \quad r(0) = r_0$$

Many interest rate models have some form of mean reversion, reverting the generated interest rate paths to some "long-run" level. Without reversion, interest rates could obtain unreasonably high and low levels. Volatility, over time, would theoretically approach infinity. Similarly, a large percentage volatility assumption would result in greater fluctuations in yield, which in turn results in a greater probability of the opportunity to refinance. The increased probability in refinancing is a greater value attributed to the implied call option, and a higher resulting option cost.[17]

In addition to a more realistic term structure, we need to expand our prepayment model to reflect the effects of multiple factors that impact prepayment. We can utilize the Richard and Roll (1989) prepayment model, which is based on empirical estimation of the mortgagor's financing condition. The model tries to explain prepayments by observing actual prepayments and relating them to the measurable factors suggested by their economic theory of prepayments. The prepayment model makes a few assumptions. The maximum CPR is 50% and the minimum CPR is 0%. The midpoint CPR at 25% occurs at a WAC-refinance rate differential at 200 basis points. At midpoint, the maximum slope is 6% CPR for a 10 basis point rate shift.

The Richard and Roll (1989) model identifies four factors that should be included in any prepayment model:[18]

1. Refinancing incentive: borrower's incentive to refinance

$$RI(t) = a + b(\arctan(c + d(WAC - r_t)))$$

where[19]

$$a = (\max \mathit{CPR} + \min \mathit{CPR})/2$$
$$b = 100(\max \mathit{CPR} - a)/(\pi/2)$$
$$d = \max \mathit{slope}/b$$
$$c = -d \; x \; \mathit{midpoint} \text{ diff.}$$

2. Seasoning (age of the mortgage):

$$\mathit{Age}(t) = \min(\frac{t}{30}, 1)$$

3. Seasonality (monthly multiplier): yearly trends in housing turnover[20]

$$\mathit{MM}(t) = (0.94, 0.76, 0.74, 0.95, 0.98, 0.92, 0.98, 1.10, 1.18, 1.22, 1.23, 0.98)$$

where t is the tth month, $t = 1...12$.

4. Burnout multiplier: A spike in refinancing due to incentives is followed by a burnout

$$\mathit{BM}(t) = 0.3 + 0.7\frac{B(t)}{B(0)}$$

where $B(t)$ is the mortgage balance at time t and $B(0)$ is the initial mortgage pool balance.

The annualized prepayment rate, $\mathit{CPR}(t)$, is equal to

$$\mathit{CPR}(t) = \mathit{RI}(t)x\mathit{Age}(t)x\mathit{MM}(t)x\mathit{BM}(t).$$

The cash flows for the MBS under this expanded prepayment model are as follows:

- $\mathit{MP}(t)$ is the scheduled mortgage payment for period (month) t:

$$\mathit{MP}(t) = B(t)\left(\frac{\mathit{WAC}/12}{1-(1+\mathit{WAC}/12)^{-\mathit{WAM}+t}}\right)$$

- $\mathit{IP}(t)$ is the interest payment for period t:

$$\mathit{IP}(t) = B(t)\left(\frac{\mathit{WAC}}{12}\right)$$

- $\mathit{PP}(t)$ is the principal prepayment for period t

$$\mathit{PP}(t) = \mathit{SMM}(t)(B(t) - \mathit{SP}(t))$$

where

$$\mathit{SMM}(t) = 1 - \sqrt[12]{1 - \mathit{CPR}(t)} \text{ and } \mathit{SP}(t) = \mathit{MP}(t) - \mathit{IP}(t).$$

- $SP(t)$ is the scheduled principal payment for period t, and $SMM(t)$ is the single monthly mortality-rate at time t.

The reduction in the mortgage balance for each month is given by

$$B(t+1) = B(t) - TPP(t)$$

where $TPP(t)$ is the total principal payment for period t. As before, we computed the expected cash flows at time t, $CF(t)$, of the MBS, and thus the MBS price P, using these formulas and Monte Carlo:

$$P = E^Q\left[\sum_{t=0}^{M} PV(t)\right] = E^Q\left[\sum_{t=0}^{M} df(t)CF(t)\right]$$

where

$$\begin{aligned}
PV &= \text{Present value for cash flow at time } t.\\
df(t) &= \prod_{k=1}^{t}\frac{1}{(1+r_k)} = \text{Discounting factor for time } t.\\
CF(t) &= MP(t) + PP(t) = TTP(t) + IP(t)\\
MP(t) &= SP(t) + IP(t)\\
TPP(t) &= SP(t) + PP(t).
\end{aligned}$$

Monte Carlo simulation can be used to help people like portfolio managers identify whether current MBS market prices are rich or cheap compared to their theoretical values and variances, and make potentially profitable trades to capture "mispricings" in the market compared to their "true" theoretical values. One can use the information from the simulation to estimate the average life of each path and the mean and variance from all the paths. From this, one can estimate prepayment risk.

There are two types of cash flow analysis approaches. The first, static cash flow analysis, assumes a constant PSA, while the second, vector (or dynamic) cash flow analysis assumes that the PSA changes over time. The static cash flow methodology estimates CFs based on different PSA speeds, and then calculates the yields on the CFs for prices and for different PSA speeds, assuming a constant interest rate volatility assumption. Static CF analysis is useful in determining what is a good price given the estimated yields based on PSA speeds, duration, average life, and other features of the mortgage or MBS. Table 3.8 shows static analysis for different par value, price, and PSA speeds assumptions.

Based on CF analysis, an investor would be willing to pay 90.75% of par or less for the MBS, if they required a yield of 9.76% for an MBS investment with a PSA of 165 (or equivalently for an investment with an average life of 2.93, and duration of 2.57).

Vector analysis is a more dynamic approach. Vector analysis can be used like static CF analysis to determine prices given required yields. The example at the bottom of Table 3.8 shows vector analysis in which different PSA speeds are assumed for three subperiods.

In general, a decrease in PSA will benefit longer-maturity tranches more than shorter maturity tranches. Slowing down prepayment increases the OAS for all tranches, more for

Table 3.8

Par Value	PSA Yield			Mean	Std Dev.
(Price as % of Par)	50%	100%	165%		
$44.127M (90.75)	8.37%	9.01%	9.76%	9.047%	.5681
$45.100M (92.75)	7.82%	8.31%	8.88%	8.3367%	.4330
$46.072M (94.75)	7.29%	7.63%	8.03%	7.6500%	.5234
$47.045M (96.75)	6.78%	6.97%	7.20%	6.9800%	.1717
$48.017M (98.75)	6.28%	6.34%	6.40%	6.3400%	.0490
Average Life (Years)	5.10	3.80	2.93		
Maturity	9.40	7.15	5.40		
Duration	4.12	3.22	2.57		
Vector Analysis:					
Month:	PSA				
1–36	50	100	165		
37–138	200	200	400		
139–357	300	300	400		
At $48.127M:					
Yield	6.02%	6.01%	6.00%	6.0100%	.00816
Average Life	3.51	2.71	2.63		
Duration	2.97	2.40	2.34		

Source: Johnson, S. (2004)

those tranches trading above par, as well as increases their price. However, changes in price are not as great for shorter duration tranches, as their prices do not move as much from a change in OAS as a longer duration tranche. Conversely, an increase in PSA will reduce the OAS and price of all tranches, especially if they are trading above par. Interest-only (IO) tranches and IO types of tranches will be adversely affected by an increase as well. A reduction in interest rate volatility increases the OAS and price of all tranches, though most of the increase is realized by the longer maturity tranches. The OAS gain for each of the tranches follows more or less the OAS durations of those tranches.[21] An increase in interest rate volatility will distribute the collateral's loss such that the longer the tranche duration, the greater the loss.

As part of the valuation model, option-adjusted duration and option-adjusted convexity are important measures. In general, duration measures the price sensitivity of a bond to a small change in interest rates. Duration can be interpreted as the approximate percentage change in price for 100-basis point parallel shift in the yield curve.[22]

For example, if a bond's duration is 3.4, this suggests that a 100-basis point increase in rates will result in a price decrease of approximately 3.4%. A 50-basis point increase

in yields will decrease the price by roughly 1.7%. The smaller the basis point change, the better the approximated change will be.

The effective duration of an MBS (or any fixed-income security) can be approximated as follows:

$$\textit{Effective Duration} = \frac{V_- - V_+}{2V_0\Delta r} \tag{3.11}$$

where

V_- = Price if yield is decreased (per \$100 of par value) by Δr.

V_+ = Price if yield is increased (per \$100 of par value) by Δr.

V_0 = Initial price (per \$100 of par value).

Δr = Number of basis points change in rates used in calculate V_- and V_+.

Effective duration—in contrast to modified duration, which is the standard measure of duration—assumes that prices in the formula (3.11) are computed assuming cash flow changes when interest rates change. Modified duration, on the other hand, assumes that if interest rates change, the cash flow does not change so that modified duration is an appropriate measure for option-free securities like Treasury bonds, but not for securities with embedded options like MBSs, where cash flows are affected by rate changes. Consequently, MBSs use effective duration, also known as OAS duration, which can be computed using an OAS model as follows. First, the bond's OAS is found using the current term structure of interest rates. Next, the bond is reprised holding OAS constant, but shifting the term structure twice—one shift increases yields and one shift decreases yields generating two prices, V_- and V_+, respectively.[23]

Subsequently, effective duration can be used with a binomial tree or with CF analysis to measure the duration of a bond with option risk or an MBS. The following steps are utilized for using a binomial tree to value a bond with an embedded option. First, take a yield curve estimated with bootstrapping and value the bond, V_0, using the calibration approach. Then, let the estimated yield curve with bootstrapping decrease by a small amount and then estimate the price of the bond using the calibration approach: V_-. Let the estimated yield curve with bootstrapping decrease by a small amount, and then estimate the price of the bond using the calibration approach: V_+. Finally, calculate the effective duration in (3.11). Similarly, using static cash flow analysis, you can calculate effective duration as follows. For a given PSA, determine the prices associated with small yield changes (you can also use a model in which you assume PSA changes as rates change), and then use the formula (3.11). It is important to note that effective duration assumes only parallel shifts in the term structure and will not correctly predict the bond price change if shifts are not parallel.

Convexity is a measure of a security that is the approximate change in price that is not explained by duration. It can be viewed as the second-order term of the Taylor expansion of the bond price as a function of yield. Bonds with positive convexity will have a greater percentage increase in price than the percentage price decrease if the yield changes by a given number of basis points. Conversely, bonds with negative convexity will have a greater percentage price decrease than percentage price increase if yields change a given number of basis points. Although positive convexity is a desirable feature of a bond, a pass-through security can exhibit either positive or negative convexity, depending on the current

mortgage refinancing rate relative to the rate on the underlying mortgage loans. Convexity can be computed as:

$$\frac{V_+ + V_- - 2(V_0)}{2V_0(\Delta r)^2} \tag{3.12}$$

If cash flows do not change when yields change, then the resulting convexity from (3.12) is a good approximation to the standard convexity of an option-free bond. However, if prices in (3.12) are derived by changing the cash flows change (by changing prepayment rates) when yields change, the resulting convexity is called *effective convexity*.[24] If prices are obtained by simulating the OAS via Monte Carlo simulation or by an OAS model, the resulting value is known as OAS convexity.

As an example of computing duration and convexity, consider a PSA 165 MBS with the following prices and yields shown in Table 3.9.

Table 3.9

Price	Yield
102.1875	6.75%
100.2813	7.00%
98.4063	7.25%

From (3.12), we find the duration is

$$\frac{102.1875 - 98.4063}{2(100.2813)(.0025)} = 7.54$$

and the convexity is

$$\frac{102.1875 + 98.4063 - 2(100.2813)}{2(100.2813)(.0025)^2} = 24$$

Thus, for a 25% change in the yield, the bond price will change by 7.54%, with a positive convexity of 24%—meaning 24% of the price change is not captured by the duration.

Figure 3.4 shows the simulated cash flows for a 30-year MBS with a 8.5% coupon on a $1,000,000 pool.

For MBS-pricing models where the underlying factor follows a diffusion process, see Kariya and Kobayashi (2000) for a one-factor (interest rate) valuation model and Kariya, Ushiyama, and Pliska (2002) for a three-factor (interest rate, mortgage rate, and housing price) valuation model.[25]

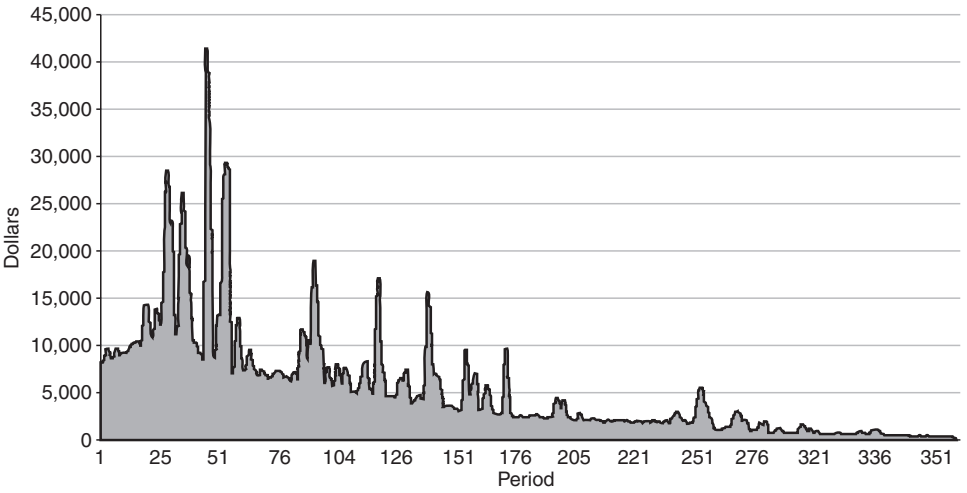

Figure 3.4 Simulated cash flows for a 30-year MBS. *Source: Bandic, I. (2002), pg. 23.*

3.6 MATLAB FIXED-INCOME TOOLKIT FOR MBS VALUATION

The Matlab Fixed-Income Toolkit can be used for MBS valuation and for computing many MBS measures, such as effective duration, convexity, and OAS. The following variables are inputs in MBS valuation in Matlab:

- **Price:** Clean price for every $100 of face value.
- **Yield:** Mortgage yield, compounded monthly (in decimal).
- **Settle:** Settlement date. A serial date or date string. Settle must be earlier than or equal to Maturity.
- **OriginalBalance:** Original balance value in dollars (balance at the beginning of each TermRemaining).
- **TermRemaining:** (Optional) Number of full months between settlement and maturity.
- **Maturity:** Maturity date. A serial date number of date string.
- **IssueDate:** Issue date. A serial date number or date string.
- **GrossRate:** Gross coupon rate (including fees), in decimal. Equal to WAC.
- **CouponRate:** Net coupon rate, in decimal. Default = GrossRate. Equal to PT rate.
- **Delay:** (Optional) Delay (in days) between payment from homeowner and receipt by bondholder. Default = 0 (no delay between payment and receipt).

- **NMBS:** Number of mortgage-backed securities.
- **PrepaySpeed:** (Optional) Relation of the conditional prepayment rate (CPR) to the benchmark model. Default = 0. Set PrepaySpeed to [] if you input a customized prepayment matrix.
- **PrepayMatrix:** (Optional) Used only when PrepayModel and PrepaySpeed are unspecified. Customized prepayment vector: A NaN-padded matrix of size max(TermRemaining)-by-NMBS. Each column corresponds to each MBS, each row corresponds to each month after settlement.
- **ZeroMatrix:** A matrix of three columns. Column 1: serial date numbers. Column 2: spot rates with maturities corresponding to the dates in Column 1. Column 3: Compounding of rates in Column 1. Values are 1 (annual), 2 (semiannual), 3 (three times per year), 4 (quarterly), 6 (bimonthly), 12 (monthly), and –1 (continuous).
- **Interpolation:** Interpolation method. Computes the corresponding spot rates for the bond's cash flow. Available methods are (0) nearest, (1) linear, and (2) cubic spline. Default = 1.

All inputs (except PrepayMatrix and ZeroMatrix) are NMBS x 1 vectors. The following variables are inputs for pricing bonds, which can in turn be used to find the implied yield curve for pricing mortgage-backed securities:

- **Face:** (Optional) Face value of each bond in the portfolio. Default = 100.
- **Yield:** Scalar or vector containing yield to maturity of instruments.
- **Settle:** Settlement date. A scalar or vector of serial date numbers. Settle must be earlier than or equal to Maturity.
- **Maturity:** Maturity date. A scalar or vector of serial date numbers of date strings.
- **ConvDates:** Conversion dates for the bonds. A matrix of serial date numbers.
- **CouponRates**: Matrix containing coupon rates for each bond in the portfolio in decimal form. The first column of this matrix contains rates applicable between Settle and dates in the first column of ConvDates.
- **Period:** (Optional) Number of coupons per year of the bond. A vector of integers. Allowed values are 0, 1, 2, 3, 4, 6, and 12. Default = 2 (semiannual).
- **Basis:** (Optional) Day count basis of the instrument. A vector of integers. 0 = actual/actual (default), 1 = 30/360, 2 = actual/360, 3 = actual/365, 4 = 30/360 (PSA compliant), 5 = 30/360 (ISDA compliant), 6 = 30/360 (European), and 7 = actual/365 (Japanese).
- **EndMonthRule:** (Optional) End-of-month rule. A vector. This rule applies only when Maturity is an end-of-month date for a month having 30 or fewer days. 0 = ignore rule, meaning that a bond's coupon payment date is always the same numerical day of the month. 1 = set rule on (default), meaning that a bond's coupon payment date is always the last actual day of the month.

Suppose we want to compute the cash flows and balances of an FHLMC mortgage pool with an initial balance of $10,000,000, PSA of 150, WAC = 8.125%, term of 360 months, and a remaining term of 357 months (the pool has been "seasoned" for three months). We use the following Matlab code:

```
OriginalBalance = 1000000;
GrossRate = 0.08125;
OriginalTerm = 360;
TermRemaining = 357;
PrepaySpeed = 125;
[Balance, Payment, Principal, Interest, Prepayment] =
  mbspassthrough(OriginalBalance,...
 GrossRate, OriginalTerm, TermRemaining, PrepaySpeed)
```

This code produces the output shown in Table 3.10 (Balance, Payment, Principal, Interest, and Prepayment).

Table 3.10

Month	Balance	Payment	Principal	Interest	Prepayment
1	998,490.00	7439.7	668.8373	6770.8	836.6
2	996,780.00	7433.4	672.8021	6760.6	1045.4
3	994,850.00	7425.7	676.6479	6749	1253.8
4	992,700.00	7416.3	680.3719	6735.9	1461.6
5	990,350.00	7405.4	683.9716	6721.4	1668.7
6	987,790.00	7392.9	687.4443	6705.5	1875
7	985,020.00	7378.9	690.7876	6688.2	2080.4
8	982,040.00	7363.4	693.9991	6669.4	2284.7
9	978,850.00	7346.3	697.0763	6649.2	2487.7
10	975,460.00	7327.7	700.017	6627.7	2689.5
350	5,640.00	832.5	788.7487	43.8	36.7
351	4,820.00	827.1	788.9469	38.2	31.4
352	4,000.00	821.8	789.1451	32.6	26.1
353	3,190.00	816.4	789.3434	27.1	20.8
355	1,590.00	805.9	789.7401	16.2	10.3
356	790.00	800.7	789.9385	10.7	5.2
357	0.00	795.5	790.137	5.3	0

Source: Johnson, S. (2004)

These values are computed in the same way as in §3.2. Given a portfolio of mortgage-backed securities, we could compute the clean prices and accrued interest using the Matlab `mbsprice` function. Suppose the yield on the portfolio is 7.25%, the WAC (gross coupon) is 8.5%, the maturity is January 10, 2034, the issue date is January 10, 2004, and we want the price on five settlement dates: March 10, 2004; May 17, 2004; May 17, 2005; January 10, 2006; and June 10, 2006 with PT (coupon) rates of 7.5%, 7.875%, 7.75%, 7.95%, and 8.125%, on each of the MBS securities, respectively. Assume the delay in the start of payments is 20 days:

```
% MBS.m  : compute MBS prices and accrued interest
Yield = 0.0725;
Settle = datenum(['10-Mar-2004';'17-May-2004';'17-May-2005';'10-Jan-
2006';'10-Jun-2006']);
Maturity = datenum('10-Jan-2034');
IssueDate = datenum('10-Jan-2004');
GrossCoupon = 0.085;
CouponRate = [0.075; 0.07875; 0.0775; 0.0795; 0.08125];
Delay = 20;
Speed = 150;
[Price, Accrt] = mbsprice(Yield, Settle, Maturity, IssueDate, ...
        GrossRate, CouponRate, Delay, Speed)
```

Table 3.11 shows the prices and accrued interest at each of the settlement dates.

Table 3.11

Settlement Date	Price	Accrued Interest
March 10, 2004	101.0937	0.0000
May 17, 2004	103.2801	0.1531
May 17, 2005	102.3677	0.1507
Jan 10, 2006	103.3897	0.0000
June 10, 2006	104.3008	0.0000

Suppose that we want to compute the OAS of the mortgage pool at the March 10, 2004 settlement date with a roughly a 28-year WAM remaining, given assumptions of a 100, 150, and 200 PSA speeds using the computed price and coupon on March 10, 2004 for the preceding mortgage pool. In Matlab, we first need to create a zero matrix constructed (implied) by bond prices (assume all bonds pay semiannual coupons) and yields:

```
Bonds = [datenum('11/21/2004')  0.045   100   2   3   1;
         datenum('02/20/2005')  0.0475  100   2   3   1;
         datenum('07/31/2007')  0.0500  100   2   3   1;
         datenum('08/15/2010')  0.0550  100   2   3   1;
         datenum('03/15/2012')  0.0575  100   2   3   1;
         datenum('02/15/2015')  0.0600  100   2   3   1;
         datenum('03/31/2020')  0.0650  100   2   3   1;
         datenum('08/15/2025')  0.0720  100   2   3   1;
         datenum('07/20/2034')  0.0850  100   2   3   1];

Yields = [0.0421; 0.0452;  0.0482; 0.0510;  0.0532;  0.0559;
0.0620; 0.0682; 0.0785];
% Since the above is Treasury data and not "selected" agency data, an
% ad-hoc method of altering the yield has been chosen for demonstration
% purposes
Yields = Yields + 0.025*(1./[1:9]');

% Get parameters from Bonds matrix
Settle = datenum('10-Mar-2004');
Maturity     = Bonds(:,1);
CouponRate   = Bonds(:,2);
Face         = Bonds(:,3);
```

```
Period       = Bonds(:,4);
Basis        = Bonds(:,5);
EndMonthRule = Bonds(:,6);

% compute bond prices
[Prices, AccruedInterest] = bndprice(Yields, CouponRate, ...
  Settle, Maturity, Period, Basis, EndMonthRule, [], [], [], [], ...
  Face);

% uses the bootstrap method to return a zero curve given a portfolio of
% coupon bonds and their prices
[ZeroRatesP, CurveDatesP] = zbtprice(Bonds, Prices, Settle);
SpotCompounding = 2*ones(size(ZeroRatesP));
ZeroMatrix = [CurveDatesP, ZeroRatesP, SpotCompounding];
Maturity = datenum('10-Jan-2034');
IssueDate = datenum('10-Jan-2004');
GrossRate = 0.085;
Delay = 20;
Interpolation = 1;
PrepaySpeed = [100 150 200];
Price = 101.0937;
CouponRate = 0.075;
Settle = datenum('10-Mar-2004');

OAS = mbsprice2oas(ZeroMatrix, Price, Settle, Maturity, ...
       IssueDate, GrossRate, CouponRate, Delay, Interpolation, ...
       PrepaySpeed)
```

The OAS results are shown in Table 3.12

Table 3.12

PSA	OAS
100	82.6670
150	101.8518
200	114.6088

We can compute the effective duration and convexity of the mortgage pool using the functions `mbsdurp` (duration given price), `mbsdury` (duration given yield), `mbsconvp` (convexity given price), and `mbsconvy` (convexity given yield). For instance, continuing with the example, we can compute the yearly duration, modified duration, and convexity of the pool on March 10, 2004 for the 100, 150, and 200 PSA speed assumptions by making the function calls following the code:

```
% compute regular duration and modified duration
[YearDuration, ModDuration] = mbsdurp(Price, Settle, Maturity,
  IssueDate, GrossRate,   CouponRate, Delay, PrepaySpeed)

% compute convexity
Convexity = mbsconvp(Price, Settle, Maturity, IssueDate, GrossRate,
  CouponRate, Delay,PrepaySpeed)
```

Table 3.13 shows duration, modified duration, and convexity for 100, 150, and 200 PSAs.

Table 3.13

PSA	Year Duration	Mod. Duration	Convexity
100	7.0669	6.8148	82.3230
150	6.1048	5.8881	62.2476
200	5.3712	5.1814	48.3368

3.7 Collateralized Mortgage Obligations (CMOs)

CMOs are securities backed by a pool of mortgages, MBSs, stripped MBSs, or CMOs. They are structured so that there are several classes of bonds; these classes are called tranches. Each tranche has a different priority claim on the principal. There are two general types of CMOs: sequential-pay tranches and planned and amortization class (PAC). In a sequential-pay tranche, each bond class is prioritized in terms of the order of the principal payment. Principal for each tranche is paid sequentially by priority: The first priority tranche's principal is paid entirely (retired) before the next class, which has its principal paid before the next class, and so on. This process continues until all the tranches in the structure are paid off. In general, the YTM of the first tranche is the lowest because it has the shortest average life and the least prepayment risk. Each successive tranche has a longer average life and a higher YTM.

The last tranche in many plain vanilla structures does not receive interest until all the tranches with shorter maturities are paid off. These classes are known as accrual bonds or "Z bonds" due to their similarity to zero-coupon bonds. The interest that would be paid to the Z bond is used to pay the principal in the shorter maturity tranches, which shortens their average lives.

Figure 3.5 shows a hypothetical distribution of principal and interest cash flows between the sequential pay bonds and the Z bond.

Suppose we form sequential-pay tranches from the mortgage portfolio described in Table 3.14: \$100M mortgage portfolio, WAM = 357 months, WAC = 8.125%, PT rate = 7.5%, and prepayment speed = 165.

The distribution of cash flows is made as follows. Principal payments are first made to A, then to B, then to C, and so on down to the residual tranches. Principal payment includes both scheduled principal payment and prepaid principal. The coupon payment is based on the remaining balance in the tranche. Table 3.15 shows the structured tranche payments.

To attract certain types of investors, floating-rate and inverse floating-rate tranches are created. These two tranches can be created from an existing one such as the C tranche. For example, a floating rate class (FR) and inverse floating rate class (IFR) in C can be constructed such that the floating rate is, for example, LIBOR + 50 basis points, and the

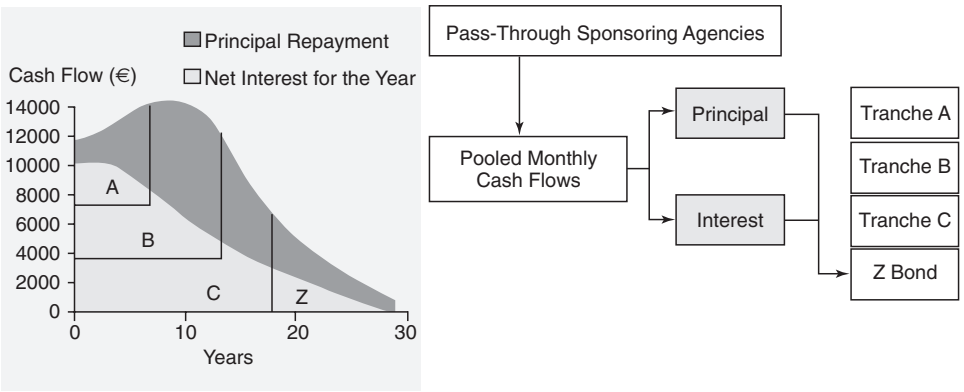

Figure 3.5 Sequential Pay Bonds and Z Bonds.

Table 3.14

Tranche	Par	PT Rate
A	$48.625M	7.5%
B	$9M	7.5%
C	$42.375M	7.5%

inverse floating rate is $28.5 - 3$ (LIBOR). The FR class is 75% of tranche C, and the IFR class is 25%. Thus, the tranches can be constructed as shown in Table 3.16.

Note that in the inverse floating rate equations, 28.5 is referred to as the cap rate (K) and 3 is called the leverage rate (L). Because the floating and inverse floating classes were created from class C, which paid a rate of 7.5%, the K and L were found such that:

$$.75[\mathit{LIBOR} + .5\%] + .25[K - L(\mathit{PLIBOR}) = 7.5\%$$

If LIBOR is 6%, then:

$$\begin{aligned} \mathit{FR} &= 6\% + .5\% = 6.5\% \\ \mathit{IFR} &= 28.5 - 3(6\%) = 10.5 \\ .75(6.5\%) &+ .25(10.5\%) = 7.5\% \end{aligned}$$

CMOs often have tranches with different rates. Such CMOs often include a special type of tranche known as a notional interest-only class, which receives only the residual interest. Notional IO classes are often described as paying a certain base interest on a notional principal. Consider the following CMO with an NIO shown in Table 3.17.

The notional principal of the NIO class is $13.75M. The interest-only (IO) class receives the excess interest of 7.5% over the rate paid on each class. For example, from class A, the IO class would receive 1.5% (7.5% − 6%) on $48.625M, which is $0.729375M. Capitalizing $.729375M at 7.5% yields a notional principal of $9.725M for the IO class on class A. The sum of the notional principals for each class yields the IO's notional principal of $13.75M, as shown in Table 3.18.

Table 3.15

Month	Balance 100,000,000.00	Interest	Principal	Tranch A Begin. Bal. 48,625,000.00	48,625,000.00 Interest	Principal
1	100,000,000.00	625,000.00	177,480.87	48,625,000.00	303,906.25	177,480.87
2	99,822,519.13	623,890.74	205,473.91	48,447,519.13	302,796.99	205,473.91
3	99,617,045.22	622,606.53	233,389.97	48,242,045.22	301,512.78	233,389.97
4	99,383,655.25	621,147.85	261,205.39	48,008,655.25	300,054.10	261,205.39
5	99,122,449.86	619,515.31	288,896.53	47,747,449.86	298,421.56	288,896.53
6	98,833,553.33	617,709.71	316,439.78	47,458,553.33	296,615.96	316,439.78
80	51,965,586.84	324,784.92	512,605.38	590,586.84	3,691.17	512,605.38
81	51,452,981.45	321,581.13	508,049.13	77,981.45	487.38	77,981.45
82	50,944,932.32	318,405.83	503,532.52	0.00	0.00	0.00
83	50,441,399.80	315,258.75	499,055.22	0.00	0.00	0.00
84	49,942,344.58	312,139.65	494,616.89	0.00	0.00	0.00
85	49,447,727.69	309,048.30	490,217.17	0.00	0.00	0.00
99	42,968,082.27	268,550.51	432,494.82	0.00	0.00	0.00
100	42,535,587.45	265,847.42	428,635.94	0.00	0.00	0.00
101	42,106,951.51	263,168.45	424,810.66	0.00	0.00	0.00
102	41,682,140.85	260,513.38	421,018.70	0.00	0.00	0.00
103	41,261,122.15	257,882.01	417,259.77	0.00	0.00	0.00
104	40,843,862.38	255,274.14	413,533.58	0.00	0.00	0.00
105	40,430,328.80	252,689.56	409,839.84	0.00	0.00	0.00
106	40,020,488.96	250,128.06	406,178.29	0.00	0.00	0.00
356	74,797.79	467.49	37,597.30	0.00	0.00	0.00
357	37,200.49	232.50	37,200.49	0.00	0.00	0.00

Tranche B Beginning Bal.	9000000 Principal	Interest	Tranche C Month	C: Beg. Bal Cum Int	Principal	Interest
9000000				42375000	0	
9000000	0	56250	1	42375000	0	0
9000000	0	56250	2	42639843.8	0	0
9000000	0	56250	3	42904687.5	0	0
9000000	0	56250	4	43169531.3	0	0
9000000	0	56250	5	43434375	0	0
9000000	0	56250	54	56411718.8	0	0
9000000	196997.83	56250	55	56676562.5	0	0
8803002.17	899641.259	55018.764	56	56941406.3	0	0
7903360.91	894022.233	49396.006	57	57206250	0	0
904465.071	850793.615	5652.9067	65	59325000	0	0
53671.45651	53671.4565	335.4466	66	59589843.8	527084	372436.52
0	0	0	67	59062759.6	575606	369142.25
0	0	0	68	58487153.2	570502	365544.71
0	0	0	69	57916651	565442	361979.07
0	0	0	356	74797.7939	37597.3	467.48621
0	0	0	357	37200.4934	37200.5	232.50308

Source: Johnson, S. (2004)

Table 3.16

Tranche	Par	PT Rate
A	\$48.625M	7.5%
B	\$9M	7.5%
FR	\$31.782M	LIBOR + 50 bps
IFR	\$10.549M	28.5 − 3 (LIBOR)

Table 3.17

Tranche	Par	PT Rate
A	\$48.625M	6.0%
B	\$9M	6.5%
Z	\$42.375M	7.0%
NIO	\$13.750M	7.5%

Table 3.18

Tranche	Par	PT Rate	(.075−PT Rate) Par	Notional Principal
A	\$48.625M	6.0%	\$729,375	\$9.725M
B	\$9M	6.5%	\$90,000	\$1.2M
Z	\$42.375M	7.0%	\$211,875	\$2.825M
			Total	\$13.75M

Suppose we have a \$30,000,000 FHLMC mortgage pool with three tranches, A, B, and C, each with a size of \$10,000,000. Assume the first tranche pool "balloons out" in 60 months, the second pool "balloons out" in 90 months, and the third is regularly amortized to maturity. The prepayment speeds are assumed to be 100, 165, and 200 for each tranche, respectively. Suppose that the delay before the first pass-through payment made after issue is 30 days, the WAC (*GrossRate* in Matlab) is 8.125%, the PT rate (*CouponRate* in Matlab) is 7.5%, the issue date is March 1, 2004, the settlement date is March 1, 2004, and the maturity is March 1, 2034. The following Matlab code computes cash flows between settle and maturity dates, the corresponding time factors in months from settle, and the mortgage pool factor (the fraction of loan principal outstanding) for each tranche:

```
% mbsfamounts
% [output] CFlowAmounts: vector of cash flows starting from Settle
% through end of the last month (Maturity)
%                CflowDates: indicates when cash flows occur, including
% at Settle.  A negative number at Settle indicates
% accrued interest is due.
%                TFactors: vector of times in months from Settle,
% corresponding to each cash flow.
%                Factors: vector of mortgage factors (the fraction of
% the balance still outstanding at the end of each month).
Settle = [datenum('1-Mar-2004');
              datenum('1-Mar-2004');
              datenum('1-Mar-2004')];
```

```
Maturity = [datenum('1-Mar-2034')];
IssueDate = datenum('1-Mar-2004');
GrossRate = 0.08125;
CouponRate = 0.075;
Delay = 30;
PSASpeed = [100; 165; 200];
[CPR, SMM] = psaspeed2rate(PSASpeed);
PrepayMatrix = ones(360,3);
PrepayMatrix(1:60,1) = SMM(1:60,1);
PrepayMatrix(1:90,2) = SMM(1:90,2);
PrepayMatrix(:,3)=SMM(:,3)
 [CFlowAmounts, TFactors, Factors] = mbscfamounts(Settle, Maturity,
     IssueDate, ...
   GrossRate, CouponRate, Delay, [], PrepayMatrix)
```

The cash flows for the difference sequential tranches are shown in Table 3.19.

We can compute the price and accrued interest of each of the mortgage pools by using the following code:

```
[Price, AccrInt] = mbsprice(Yield, Settle, Maturity, IssueDate,
GrossRate, CouponRate, Delay, PrepaySpeed, PrepayMatrix)
```

Table 3.19

Month	Tranche A CF	Tranche B CF	Tranche C CF
1	0	0	0
2	70,708.00	71,794.00	72,379.00
3	72,368.00	74,534.00	75,702.00
4	74,015.00	77,256.00	79,004.00
5	75,649.00	79,957.00	82,283.00
59	95,117.00	105,150.00	108,680.00
60	94,592.00	104,190.00	107,470.00
61	94,070.00	103,240.00	106,270.00
62	7,580,500.00	102,290.00	105,080.00
63	0.00	101,350.00	103,910.00
64	0.00	100,420.00	102,750.00
88	0.00	80,383.00	78,340.00
89	0.00	79,637.00	77,455.00
90	0.00	78,897.00	76,579.00
91	0.00	78,163.00	75,713.00
92	0.00	4,811,500.00	74,857.00
93	0.00	0.00	74,009.00
358	0.00	0.00	2,017.40
359	0.00	0.00	1,976.90
360	0.00	0.00	1,936.80
361	0.00	0.00	1,897.30

The price and accrued interest is shown in Table 3.20. No tranche pool has any accrued interest because the settlement date is the same as the issue date.

Table 3.20

	Price	Accrued Interest
Tranche A	101.1477	0
Tranche B	100.8520	0
Tranche C	100.7311	0

We can compute the effective duration and convexity of the mortgage pool using the `mbsdurp` (duration given price), `mbsdury` (duration given yield), `mbsconvp` (convexity given price), and `mbsconvy` (convexity given yield). For instance, continuing with the example, we can compute the yearly duration, modified duration, and convexity of the pool on March 10, 2004 for the 100, 150, and 200 PSA speed assumptions by making the function calls following the code:

```
% compute regular duration and modified duration
[YearDuration, ModDuration] = mbsdurp(Price, Settle, Maturity,
IssueDate, GrossRate,
   CouponRate, Delay, PrepaySpeed)

% compute convexity
Convexity = mbsconvp(Price, Settle, Maturity, IssueDate, GrossRate,
CouponRate, Delay,
    PrepaySpeed)
```

The duration, modified duration, and convexity results returned from the Matlab functions above for 100, 150, and 200 PSAs are shown in Table 3.21. Figure 3.6 shows simulated cash flows for a sequential-pay CMO.

Table 3.21

PSA	Year Duration	Mod. Duration	Convexity
100	7.0669	6.8148	82.3230
150	6.1048	5.8881	62.2476
200	5.3712	5.1814	48.3368

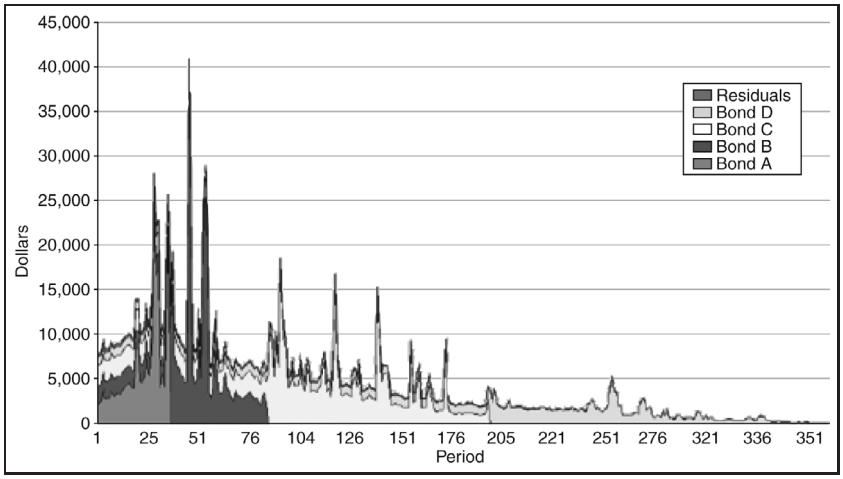

Figure 3.6 Simulated CMO Cash Flows. *Source: Bandic (2002), 28.*

3.8 CMO Implementation in C++

CMO.h

```
#ifndef _CMO__
#define _CMO__

#include <vector>
#include <algorithm>
#include <numeric>
#include "Constants.h"
#include "MBS.h"

using namespace std;

class Tranche
{
  public:
        Tranche() {}
        Tranche(char clas, double balance, double coupon)
          : initBalance_(balance), balance_(balance), coupon_(coupon),
              clas_(clas) {}
        virtual ~Tranche() {}
        double initBalance_;
        double balance_;
        double coupon_;
        vector<double> cashFlows_;
        vector<double> sumCF_;
        vector<double> inter_;
        vector<double> principal_;
        vector<double> discount_;
        vector<double> T_;
        double price_;
```

```
        double interest_;
        double princip_;
        double averageLife_;
        char clas_;
};

class CMO
{
  public:
        CMO(MBS m, vector<Tranche> tr) : mbs(m), tranche(tr)
        {
          for (int i = 0; i < tranche.size(); i++)
                collateral_.push_back(tranche[i].balance_);
        }
        virtual ~CMO() { }
        void calcTrancheCF();
        inline double calcCPR(double SMM) { return 100*(1-pow((1-
           (SMM/100)),12)); }
        inline double calcSMM(double scheduleBal, double actualBal) {
          return 100*((double)(scheduleBal - actualBal)/scheduleBal);
        }
        inline double calcPSA(double age, double CPR) {

          return 100*((double)(CPR/(0.2*min(age,30))));
    }
    inline double calcRefinance(double r) {

          double WAC = mbs.getWAC();
          double a = (double) 0.5/2;
          double b = (double) 100*((0.5 - a)/(PI/2));
          double d = (double) 0.06/b;
          double c = (double) -d*0.02;
          return (double) (a + b*(atan(c + d*(WAC - r))));
        }
        inline double calcBurnout(int t, Tranche tr, double balance) {

          return (double) (0.3 + 0.7*((double)balance/1000000));
        }
        inline double calcMP(int t, Tranche tr, double balance)  {

          double WAC = mbs.getWAC();
          double WAM = mbs.getWAM();

          return balance*(((double)WAC/12)/(1-pow((1+(double)WAC/12),
             -WAM+t)));
        }
        inline double calcIP(int t, Tranche tr, double r,
           double balance) {

          double WAC = mbs.getWAC();

          return (balance)*((double)(tr.coupon_/12));
        }
        inline double calcPP(int t, Tranche tr, double r,
           double balance) {

          double SMM = calcSMM1(t,tr,r,balance);
```

```
          double SP = calcSP(t,tr,r,balance);

          return SMM*(balance - SP);

        }
        inline double calcMM(int t) {

          double MM[12] = { 0.94, 0.76, 0.74, 0.95, 0.98, 0.92, 0.98,
                            1.10, 1.18, 1.22, 1.23, 0.98};
          int rem =  t % 12;

          if (t == 1)
                rem = 1;

          return MM[rem-1];

        }
        inline double calcCPR1(int t, Tranche tr, double r,
           double balance) {

          double RI = calcRefinance(r);
          double age = calcAge(t);
          double MM = calcMM(t);
          double BM = calcBurnout(t,tr,balance);

          return RI*age*MM*BM;

        }
        inline double calcAge(int t) {
          return min((double)t/30,1);
        }
        inline double calcSMM1(int t,Tranche tr, double r,
           double balance) {

          double CPR = calcCPR1(t,tr,r,balance);

          return (1 - pow((1 - CPR),(double)1/12));
        }
        inline double calcSP(int t, Tranche tr, double r,
           double balance) {

          double MP = calcMP(t,tr,balance);
          double IP = calcIP(t,tr,r,balance);

          return MP - IP;
        }
        void calcCashFlows(double initRate, double financeRate, int N,
           int M);
  private:
        MBS mbs;
        vector<Tranche> tranche;
        vector<double> collateral_;
};

#endif
```

Here are the method definitions:

CMO.cpp

```
#include "CMO.h"
#include "Utility.h"

void CMO::calcCashFlows(double initRate, double financeRate, int N, int M)
{
   Utility util;
   int i, t = 0;
   double r = 0.0715;
   const double kappa = 0.29368;
   const double vol = 0.11;
   const double theta = 0.08;
   double deviate = 0;
   long seed = 0;
   long* idum = 0;
   double balance = 0;
   double sum = 0;
   double S[4] = {0};
   double sum1 = 0;
   double sum2 = 0;
   double sum3 = 0;
   double sumA = 0;
   double sumB = 0;
   double sumC = 0;
   double sumD = 0;
   double CPR = 0;
   double interest = 0.0;
   double mbsPrice = 0;
   double stdErr = 0;
   double stdDev = 0;
   double totalsum = 0;
   double totalsumA = 0;
   double totalsumB = 0;
   double totalsumC = 0;
   double totalsumD = 0;
   double totalsum2 = 0;
   double schedulePrincipal = 0;
   double prepaidPrincipal = 0;
   double discount = 0;
   double principal = 0;
   double pay = 0.0;
   double r1 = 0.0;
   double rr = 0.0;
   int cnt = 0;
   double trancheBal = 0.0;
   double T = mbs.getMaturity();
   double WAM = mbs.getWAM();
   double OAS = mbs.getOAS();
   double dt = (double) T/N;
   double interest1 = 0;
   vector<double> disc(0.0);
   vector<double> time1;
   TNT::Array1D<double> CF(SIZE_X);   // cash_flow
```

```
vector<double> p;

srand(unsigned(time(0)));
seed = (long) rand() % 100;
idum = &seed;

for (t = 1; t <= N; t++)
  time1.push_back((double)(t-1)/12);

for (i = 0; i < M; i++)
{
  r = initRate;
  sum = 0;
  sumA = 0;
  sumB = 0;
  sumC = 0;
  sumD = 0;
  schedulePrincipal = 0;
  prepaidPrincipal = 0;
  balance = 1000000;
  cnt = 1;
  disc.clear();
  disc.empty();
  disc.push_back(r);
  p.clear();
  p.push_back(0);

  for (int j = 0; j < tranche.size(); j++)
  {
    tranche[j].balance_ = collateral_[j];
    tranche[j].inter_.clear();
    tranche[j].principal_.clear();
    trancheBal = calcPP(0,tranche[j],r,tranche[j].balance_) +
       calcMP(0,tranche[j],tranche[j].balance_);
    tranche[j].principal_.push_back(trancheBal);
    tranche[j].interest_ = calcIP(0,tranche[j],r,tranche[j].balance_);
    tranche[j].inter_.push_back(tranche[j].interest_);
    S[j] = 0;
  }

  for (t = 1; t <= N; t++)
  {
    balance = balance - (schedulePrincipal + prepaidPrincipal);
    deviate = util.gasdev(idum);
    r = r + kappa*(theta - r)*dt + vol*r*sqrt(dt)*deviate;
    disc.push_back(r);
    interest = calcIP(t,tranche[cnt-1],r,balance);
    schedulePrincipal = calcMP(t,tranche[cnt-1],balance);
    prepaidPrincipal = calcPP(t,tranche[cnt-1],r,balance);
    tranche[cnt-1].balance_ = tranche[cnt-1].balance_ -
       schedulePrincipal - prepaidPrincipal;
    principal = schedulePrincipal + prepaidPrincipal;
    tranche[cnt-1].principal_.push_back(principal);
    tranche[cnt-1].princip_ = principal;
    p.push_back(principal);

    if (tranche[cnt-1].balance_ > 0)
```

```
  interest1 = calcIP(t,tranche[cnt-1],r,tranche[cnt-1].balance_);
else
  interest1 = 0;

tranche[cnt-1].inter_.push_back(interest1);
tranche[cnt-1].interest_ = interest1;

for (int k = 1; k <= tranche.size(); k++)
{
  if (k != cnt)
  {
     interest1 = calcIP(t,tranche[k-1],r,tranche[k-1].balance_);

     if (tranche[k-1].balance_ != 0 )
     {
       tranche[k-1].inter_.push_back(interest1);
       tranche[k-1].interest_ = interest1;
     }
     else
     {
       tranche[k-1].inter_.push_back(0.0);
       tranche[k-1].interest_ = 0.0;
     }

     tranche[k-1].principal_.push_back(0.0);
     tranche[k-1].princip_ = 0.0;
  }

  rr = mbs.computeZeroRates(t-1,disc);
  S[k-1] = (tranche[k-1].interest_ + tranche[k-1].princip_)/
     (pow(1+rr+OAS,(double)(t-1)/12));

  if (k == 1)
    sumA = sumA + S[k-1];
  else if (k == 2)
    sumB = sumB + S[k-1];
  else if (k == 3)
    sumC = sumC + S[k-1];
  else
    sumD = sumD + S[k-1];
}

if (tranche[cnt-1].balance_ > 0)
{
  if (balance >= schedulePrincipal)
  {
    if (t != N)
       CF[t-1] = schedulePrincipal + interest + prepaidPrincipal;
    else
       CF[t-1] = interest + balance;

    rr = mbs.computeZeroRates(t-1,disc);
    sum = sum + CF[t-1]/(pow(1+rr+OAS,(double)(t-1)/12));
  }
  else
    goto x;
}
```

```
      else
      {
        tranche[cnt-1].balance_ = 0;
        cnt++;
      }
    }
    x:
      totalsum = totalsum + sum;
      totalsumA = totalsumA + sumA;
      totalsumB = totalsumB + sumB;
      totalsumC = totalsumC + sumC;
      totalsumD = totalsumD + sumD;
      totalsum2 = totalsum2 + sum*sum;
  }

  calcTrancheCF();
  for (int j = 0; j < tranche.size(); j++)
  {
       sum1 = 0;
       sum2 = 0;
       for (i = 0; i < tranche[j].principal_.size(); i++)
       {
         sum1 = sum1 + (time1[i])*(tranche[j].principal_[i]);
         sum2 = sum2 + tranche[j].principal_[i];
       }
       tranche[j].averageLife_ = sum1/sum2;
  }
  sum1 = 0;
  sum = accumulate(p.begin(),p.end(),0);
  for (j = 0; j < p.size(); j++)
       sum1 = sum1 + time1[j]*p[j];

  std::cout << endl;
  std::cout << "collateral price = " << totalsum/M <<  " " << "Ave.Life = "
     << sum1/sum << endl;
  std::cout << "Tranche A price =  " << totalsumA/M << " " << "Ave.Life = "
     << tranche[0].averageLife_ << endl;
  std::cout << "Tranche B price =  " << totalsumB/M << " " << "Ave.Life = "
     << tranche[1].averageLife_ << endl;
  std::cout << "Tranche C price =  " << totalsumC/M << " " << "Ave.Life = "
     << tranche[2].averageLife_ << endl;
  std::cout << "Tranche Z price =  " << totalsumD/M << " " << "Ave.Life = "
     << tranche[3].averageLife_ << endl;

  T = mbs.getMaturity();
  stdDev = sqrt(totalsum2 - (double)(totalsum*totalsum)/M)*
     (exp(-2*initRate*T)/(M-1));
  stdErr = (double) stdDev/sqrt(M);
}

void CMO::calcTrancheCF()
{
  vector<Tranche>::iterator iter;
  vector<double>::iterator iter1;
  vector<double>::iterator iter2;
  int cnt = 1;
```

```
  for (iter = tranche.begin(); iter != tranche.end(); iter++)
  {
     iter2 = iter->inter_.begin();
        cnt = 1;
        for (iter1 = iter->principal_.begin(); iter1 !=
           iter->principal_.end(); iter1++)
        {
          std::cout << "Mo." << cnt << "  Class: " << iter->clas_
            << " " << "Principal= " << *iter1
            << " " << "Coupon= " << *iter2 << endl;
          iter2++;
          cnt++;
        }
  }
}
```

The main method is as follows:

Main.cpp

```
void main()
{

    std::cout.precision(7);
    double principal = 1000000;      // underlying principal notional)
                                     // of MBS
    double coupon = 0.08;            // coupon rate
    double WAC = 0.08;               // weighted average coupon rate
    double WAM = 10;                 // weighted average maturity
    double OAS = 0.02;               // option adjusted spread
    double initSpotRate = 0.06;      // spot rate
    double initRefinanceRate = 0.08; // refinance rate
    int N = 10;                      // number of time steps in tree
    long int M = 100000;             // number of simulation paths

    MBS mbs(principal,coupon,WAC,WAM,OAS);

    vector<Tranche> tranche;
    Tranche trA('A',500000,0.06);
    tranche.push_back(trA);
    Tranche trB('B',300000,0.065);
    tranche.push_back(trB);
    Tranche trC('C',200000,0.07);
    tranche.push_back(trC);
    Tranche trZ('Z',100000, 0.075);
    tranche.push_back(trZ);

    std::cout << endl;
    std::cout << "Pricing CMO Tranches..." << endl << endl;
    CMO cmo(mbs,tranche);
    cmo.calcCashFlows(initSpotRate,initRefinanceRate,N,M);

}
```

The output is as follows:

```
Pricing CMO Tranches...

Mo.1   Class: A Principal= 51851.6  Coupon= 2500
Mo.2   Class: A Principal= 115430   Coupon= 1922.85
Mo.3   Class: A Principal= 114668.5 Coupon= 1349.507
Mo.4   Class: A Principal= 113798.2 Coupon= 780.5165
Mo.5   Class: A Principal= 113024   Coupon= 215.3967
Mo.6   Class: A Principal= 111815.5 Coupon= 0
Mo.7   Class: A Principal= 0        Coupon= 0
Mo.8   Class: A Principal= 0        Coupon= 0
Mo.9   Class: A Principal= 0        Coupon= 0
Mo.10  Class: A Principal= 0        Coupon= 0
Mo.1   Class: B Principal= 31110.96 Coupon= 1625
Mo.2   Class: B Principal= 0        Coupon= 1625
Mo.3   Class: B Principal= 0        Coupon= 1625
Mo.4   Class: B Principal= 0        Coupon= 1625
Mo.5   Class: B Principal= 0        Coupon= 1625
Mo.6   Class: B Principal= 0        Coupon= 1625
Mo.7   Class: B Principal= 110373.5 Coupon= 1027.143
Mo.8   Class: B Principal= 108932.9 Coupon= 437.0903
Mo.9   Class: B Principal= 107334.2 Coupon= 0
Mo.10  Class: B Principal= 0        Coupon= 0
Mo.1   Class: C Principal= 20740.64 Coupon= 1166.667
Mo.2   Class: C Principal= 0        Coupon= 1166.667
Mo.3   Class: C Principal= 0        Coupon= 1166.667
Mo.4   Class: C Principal= 0        Coupon= 1166.667
Mo.5   Class: C Principal= 0        Coupon= 1166.667
Mo.6   Class: C Principal= 0        Coupon= 1166.667
Mo.7   Class: C Principal= 0        Coupon= 1166.667
Mo.8   Class: C Principal= 0        Coupon= 1166.667
Mo.9   Class: C Principal= 0        Coupon= 1166.667
Mo.10  Class: C Principal= 105320.6 Coupon= 552.2968
Mo.1   Class: Z Principal= 10370.32 Coupon= 625
Mo.2   Class: Z Principal= 0        Coupon= 625
Mo.3   Class: Z Principal= 0        Coupon= 625
Mo.4   Class: Z Principal= 0        Coupon= 625
Mo.5   Class: Z Principal= 0        Coupon= 625
Mo.6   Class: Z Principal= 0        Coupon= 625
Mo.7   Class: Z Principal= 0        Coupon= 625
Mo.8   Class: Z Principal= 0        Coupon= 625
Mo.9   Class: Z Principal= 0        Coupon= 625
Mo.10  Class: Z Principal= 0        Coupon= 625

collateral price = 682754.6 Ave.Life = 0.4104376
Tranche A price =  564751.1 Ave.Life = 0.2279203
Tranche B price =  321741.6 Ave.Life = 0.5318972
Tranche C price =  108757.8 Ave.Life = 0.6266037
Tranche Z price =  5460.491 Ave.Life = 0
```

3.9 PLANNED AMORTIZATION CLASSES (PACS)

Planned amortization classes (PACs) (also called planned redemption obligations) are tranches set up such that they have zero (or at least minimum) prepayment risk. PACs are set up by applying low and high PSA speeds to the collateral. The PAC bond then receives a promise of the minimum CF each month, with a support bond created that receives the rest. These support classes, sometimes referred to as companions, absorb principal payments and pay off sooner if the PSA exceeds the PAC range. If the PSA is below the range, the companion classes have a longer life and amortization schedule. In either case, the PAC classes experience less volatility than they would in a sequential-pay structure because of the stability provided by the companions.

PACs are much less sensitive to prepayment risk than standard pass-throughs as long as the PSA speed falls between the low and high PSA thresholds used. Suppose one applied 90 and 300 PSA models to the collateral of a \$100M mortgage pool with a WAC $= 8.125\%$, WAM $= 357$ months, and PT rate $= 7.5\%$. This would yield two different monthly principals over the 357-month period. We also need to factor in a seasoning factor, a factor that accounts for the prepayment based on the season, which we assume to be 3. Table 3.22 shows the cash flows for a PAC with the above features.

To show how these calculations were made, we know that in the first month (period 1), the PAC balance is \$100,000,000. The computed interest for this first month is:

$$I_1 = (0.075/12) * (100,000,000) = \$625,000$$

The computed PAC principal payment is:

$$p = \frac{(0.08125/12)(100,000,000)}{1-(1/(1+0.08125/12))^{357}} = \$743,967.06$$

The PAC scheduled principal payment is:

$$743,967.06 - (0.085/12)(100,000,000) = 66,883.73$$

The low PAC PSA speed is assumed to be 90. We compute the PAC adjuster seasoning factor, which we take to be:

$$Min\left(\frac{month + seasoning\ factor}{number\ of\ months\ until\ fixed\ CPR}, 1\right)$$

Thus, in the first month, this value is:

$$Min\left(\frac{1+3}{30}, 1\right) = 0.13333$$

The PAC adjuster seasoning factor is then applied to the PAC CPR:

$$\left(\frac{PSA}{100}\right)(Fixed\ CPR)(adjuster\ seasoning\ factor) = \left(\frac{90}{100}\right)(0.06)(0.1333)$$
$$= 0.0072$$

Table 3.22

Pac Period	Pac Low PSA Pr	Pac high PSA Pr	Pac Min. Principal	Pac Int 0.08	collateral Balance 100,000,000.00	collateral Interest	collateral Prncipal
1	127,042.38	268,982.80	127,042.38	381,088.87	100,000,000.00	625,000.00	177,480.87
2	142,460.86	319,853.03	142,460.86	380,294.85	99,822,519.13	623,890.74	205,473.91
3	157,844.28	370,548.61	157,844.28	379,404.47	99,617,045.22	622,606.53	233,389.97
4	173,185.47	420,991.54	173,185.47	378,417.95	99,383,655.25	621,147.85	261,205.39
5	188,477.28	471,103.66	188,477.28	377,335.54	99,122,449.86	619,515.31	288,896.53
6	203,712.56	520,806.83	203,712.56	376,157.55	98,833,553.33	617,709.71	316,439.78
7	218,884.17	570,023.17	218,884.17	374,884.35	98,517,113.54	615,731.96	343,811.60
102	363,181.39	371,031.65	363,181.39	129,910.54	41,682,140.85	260,513.38	421,018.70
103	361,690.25	364,654.51	361,690.25	127,640.65	41,261,122.15	257,882.01	417,259.77
104	360,206.39	358,384.81	358,384.81	125,380.09	40,843,862.38	255,274.14	413,533.58
105	358,729.77	352,220.75	352,220.75	123,140.18	40,430,328.80	252,689.56	409,839.84
106	357,260.36	346,160.58	346,160.58	120,938.80	40,020,488.96	250,128.06	406,178.29
107	355,798.12	340,202.57	340,202.57	118,775.30	39,614,310.67	247,589.44	402,548.63
356	150,750.84	2,573.92	2,573.92	31.76	74,797.79	467.49	37,597.30
357	150,372.35	2,507.45	2,507.45	15.67	37,200.49	232.50	37,200.49

Source: Johnson, S. (2004)

The SMM is:

$$SMM = (1 - (1 - PAC\ CPR)^{1/357} = (1 - (1 - 0.0072)^{1/357} = 0.00060199$$

The PAC prepaid principal is:

$$SMM(balance - scheduled\ principal) \cdot$$
$$0.00060199(100,000,000 - 66,883.73) = \$60,158.65$$

The PAC cash flow is:

$$PAC\ Interest + PAC\ Scheduled\ Principal + PAC\ Prepaid\ Principal = 625,000 + 66,883.73 + 60,148.65 = \$752,042.38$$

We can now compute the PAC low PSA total prepayment amount:

$$PAC\ Scheduled\ Principal + PAC\ Prepaid\ Principal = 66,883.72 + 60,158.65 = \$127,042.38$$

The same computations are then made for the high PSA level of 300. The main difference is that the PAC CPR in this case is:

$$\left(\frac{PSA}{100}\right)(Fixed\ CPR)(adjuster\ seasoning\ factor) = \left(\frac{300}{100}\right)(0.06)(0.1333) = 0.024$$

The PAC bond has an average life of 7.26 years. Moreover, between PSA speeds of 90 and 300, the PAC bond's average life is 7.26 years, implying no prepayment risk. Table 3.23 shows average life for the PAC and support bond for various PSA assumptions.

Table 3.23

PSA Speed	PAC	Support	Collateral
0	10.36	23.84	20.36
50	8.04	21.69	15.36
90	7.26	20.06	12.26
100	7.26	18.56	11.67
150	7.26	12.56	9.33
200	7.26	8.36	7.69
250	7.26	5.35	6.52
300	7.26	3.11	5.64
350	6.61	2.91	4.98
400	6.06	2.74	4.45

The PAC bond can be broken into other PAC tranches. The most common is a sequential-pay PAC. For example, one can form six sequential-pay PACs using the previous collateral. The average life for the PAC classes will be stable within the 90–300 PSA range; 90–300 is referred to as the *collar*. Some PACs will move outside that range. This is referred to as the *effective collar*. The more classes you have, the more narrow you make the windows, making the PAC resemble a bullet bond. Such PACs could be sold to liability-management funds to meet liabilities with certain liabilities or durations—cash flow matching. In the 1980s, one could find CMOs (especially PACs) with as many as 70 tranches; in the early 1990s, the average number of tranches was 24. Like the PACs, the support bond also can be divided into different classes: sequential-pay, floaters, accrual bonds, and so on.

Targeted amortization classes (TACs) also offer prepayment protection within a defined PSA range, but not below the PSA used to price the CMO. This could result in a lengthening of average life if prepayments slow down, and for this reason, TACs offer higher yields in relation to comparable PACs.

Figure 3.7 shows the cash flows paid to a hypothetical GNMA PAC 100/300.

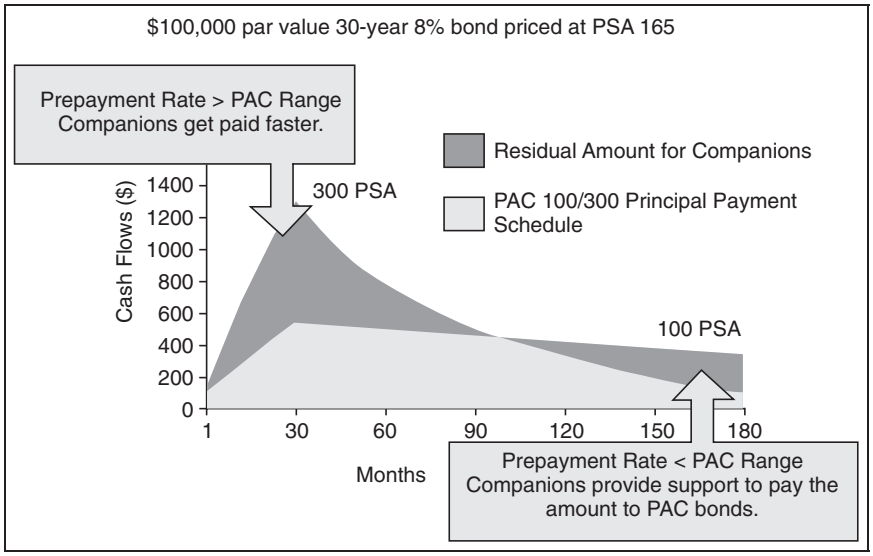

Figure 3.7 GNMA PAC 100/300 Cash Flows.

3.10 Principal- and Interest-Only Strips

Stripped MBSs were introduced by FNMA in 1986. Any MBS can be "stripped" and sold separately by directing a collateral's cash flows into principal-only (PO) or interest-only (IO) securities. IO classes receive just the interest on the mortgages. PO classes receive just the principal payments. The yield on PO bonds depends on the speed of prepayment. The faster the prepayment, the greater the yield. For instance, PO investors who paid $75 million for a mortgage portfolio with a principal of $100 million would receive a higher

yield if the \$100 million were paid early (e.g., first years) than if it were spread out. POs have an inverse price-interest rate sensitivity relationship: If interest rates decrease, then prepayments increase so that the PO (yield) return increases and its price (value) increases. Analogously, if interest rates increase, then prepayments decrease so that the PO (yield) return decreases and its price (value) decreases.

Because IO investors receive interest on the outstanding principal, they want prepayments to be slow. For example, IO investors holding an IO claim on a \$400 million 7.5% pool would receive \$30 million ($= \$400 * 0.075$) if the principal were paid immediately. By contrast, if the principal were paid off by equal increments over four years, the return would be \$75 million (see Table 3.24).

Table 3.24

Year 1	$(\$400M)(0.75) = \$30.0M$
Year 2	$(\$300M)(0.75) = \$22.5M$
Year 3	$(\$200M)(0.75) = \$15.0M$
Year 4	$(\$100M)(0.75) = \$7.5M$
Total	$\$75M$

Because a rate decrease augments speed, it lowers the return on an IO bond, causing its value to decrease. Whether IO bonds decrease in response to a rate decrease depends on whether this effect dominates the effect of lower discount rates on increasing value. In other words, when interest rates decrease, the prepayments increase, which decreases the return, and the value of the IO must be balanced against the increase in value from the effect of lower discount rates. The two effects may offset one another so that it is possible that there is a direct relationship between value and return for an IO bond.

Like CMOS, stripped MBSs are a derivative product. Both strips are extremely volatile and, as stated, dependent on prepayment rates. POs perform well in high prepayment environments when the principal purchased at a discount is returned at par, faster than expected, making them a bullish investment with a large, positive duration. IOs perform better if prepayments are slow because the principal remains outstanding for a longer period and interest payments continue, making them an investment with a negative duration—i.e., their price increases as interest rates increase and vice versa. IOs are often used to hedge interest rate risks in MBS or CMO portfolios. Portfolio losses that are caused by an increase in rates are partially or fully offset by a corresponding appreciation in the IO position, depending on the structure of the hedge.

Consider a stripped MBS with a collateral pool of \$100 million, WAM $= 357$ months, WAC $= 0.08125$, PT rate $= 0.075$, and a PSA $= 165$. The cash flows are given as shown in Table 3.25.

Table 3.25

Period	Balance 100000000	Interest	Sch. Prin.	Prepaid Prin.	Principal		Stripped MBS PO	IO
1	100000000	625000	65216.47156	110598.9911	175815.4627		175815.4627	625000
2	99824184.54	623901.153	65592.16276	138216.4665	203808.6293		203808.6293	623901.2
3	99620375.91	622627.349	65951.60553	165774.6371	231726.2426		231726.2426	622627.3
4	99388649.67	621179.06	66294.44744	193250.204	259544.6515		259544.6515	621179.1
5	99129105.01	619556.906	66620.34673	220619.862	287240.2088		287240.2088	619556.9
6	98841864.81	617761.655	66928.97288	247860.3319	314789.3047		314789.3047	617761.7
7	98527075.5	615794.222	67220.00709	274948.3929	342168.4		342168.4	615794.2
104	40930020.29	255812.627	59884.94969	353521.5406	413406.4903		413406.4903	255812.6
105	40516613.8	253228.836	59775.10019	349946.5766	409721.6768		409721.6768	253228.8
106	40106892.12	250668.076	59665.45219	346403.4839	406068.9361		406068.9361	250668.1
107	39700823.19	248130.145	59556.00532	342891.9853	402447.9906		402447.9906	248130.1
108	39298375.2	245614.845	59446.75922	339411.8057	398858.5649		398858.5649	245614.8
109	38899516.63	243121.979	59337.71351	335962.6724	395300.3859		395300.3859	243122
110	38504216.25	240651.352	59228.86783	332544.3152	391773.183		391773.183	240651.4
111	38112443.06	238202.769	59120.22181	329156.4661	388276.6879		388276.6879	238202.8
112	37724166.38	235776.04	59011.77508	325798.8595	384810.6346		384810.6346	235776
113	37339355.74	233370.973	58903.52728	322471.2322	381374.7594		381374.7594	233371
356	75665.72172	472.910761	37703.25592	328.3705641	38031.62648		38031.62648	472.9108
357	37634.09524	235.213095	37634.09524	-3.02093E-12	37634.09524		37634.09524	235.2131

3.11 Interest Rate Risk

The MBS interest rate risk is similar to that of other fixed-income securities: When interest rates fall, price goes up and vice versa. However, the prepayment optionality "embedded" in MBS impacts the degree of price movement based on the relationship between the security's coupon rate and current mortgage rates. When a pass-through coupon is either at or above current mortgage rates, homeowners are more likely to exercise the prepayment option. As the likelihood of prepayment increases, the price of the MBS pass-through does not go up as much as that of an otherwise identical security with no optionality due to the increased prepayment risk. This is known as *negative convexity*.

When a pass-through coupon is below current mortgage rates, or "out of the money," homeowners are unlikely to exercise the prepayment option. Although the prepayment option is less likely to be exercised, the price of the pass-through still exhibits negative convexity because investors maintain their principal investment at lower levels than the current market rate for longer periods of time. This is known as *extension risk*. Figure 3.8 shows the inverse relationship between the price of a regular fixed-income security and interest rates. Figure 3.9 shows the negative convexity of an MBS.

3.12 Dynamic Hedging of MBS

Institutions hold significant positions in mortgage-backed securities (MBSs) for a variety of reasons. Hedging interest rate risk of MBSs is an important concern whether the positions reflect trades on relative value or inventory holdings due to main businesses. MBSs hedging is complicated by the fact that the timing of the cash flows is dependent on the prepayment behavior of the pool. In particular, mortgagors are more likely to prepay given the incentive to refinance when interest rates fall. Thus, fixed-rate investors are implicitly writing a call option on the corresponding fixed-rate bond.[26] Though other factors influence

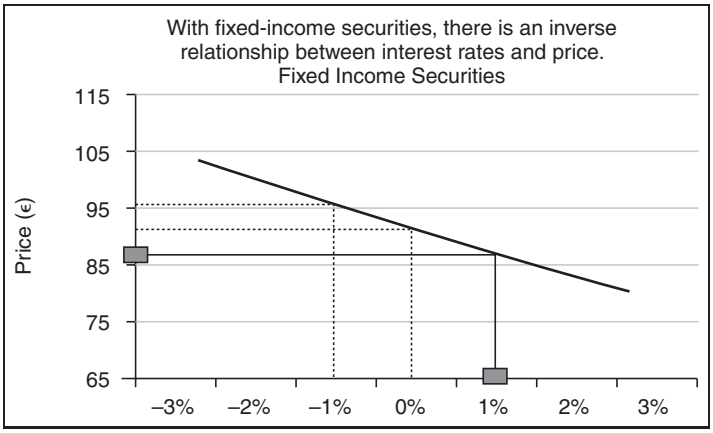

Figure 3.8 Inverse relationship between the price of a regular fixed income security and interest rates.

prepayments—e.g., seasonality and burnout—interest rates are the predominant factor in valuing MBSs. Because of this predominance, U.S. Treasury securities, or more specifically, Treasury note (T-note) futures, are often used to hedge MBSs. There are two reasons: (1) T-note futures are very liquid instruments; and (2) the prices of those instruments are determined by the underlying term structure of interest rates and thus relate directly to the value of MBSs.[27] We follow the work of Boudoukh, Richardson, Stanton, and Whitelaw (1995) in the following discussion.

There are two common approaches to hedging MBSs using T-note futures. The first is purely empirical and involves the regression of past returns on MBSs against past returns on T-note futures. The estimated regression coefficient from the resulting relation can then be used to hedge the interest rate risk of MBSs using the risk in T-notes. The advantage of

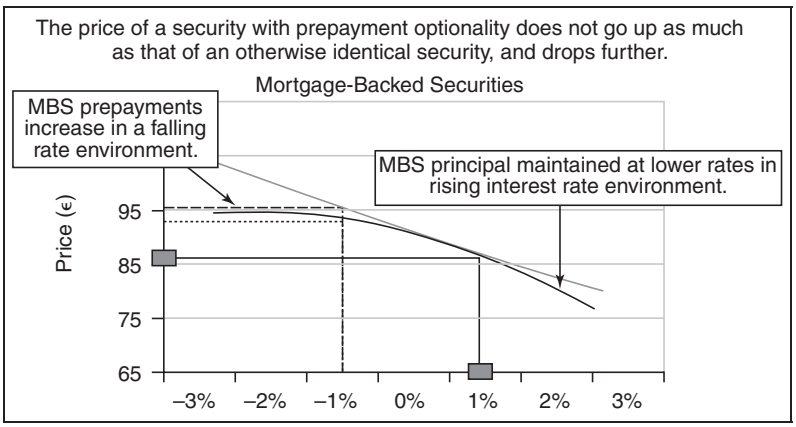

Figure 3.9 Negative convexity of a MBS.

this method is that it does not involve strong assumptions regarding the underlying model for the evolution of interest rates or prepayments.[28] The disadvantage is that the method is static in nature. It does not explicitly adjust the hedge ratio for changes in interest rates and mortgage prepayments, which can potentially be detrimental from mishedging a large portfolio exposure.[29] Consequently, the observations used in the regression represent an average of the relation between MBSs and T-note futures only over the sample period, which may or may not be representative of the current period.

As an alternative, a second approach is model-based. It involves specification of the interest rate process and a prepayment model. The assumptions then help map an MBS pricing functional to interest rates and possibly other factors.[30] The approach represents a dynamic method for determining co-movements between MBS prices and T-note futures prices. These co-movements are completely specified by conditioning on current values of the relevant economic variables and on particular parameter values. The basic idea is to estimate a conditional hedge ratio between returns on an MBS and returns on a T-note futures. This is important for MBSs because, as interest rates change, expected future prepayments change, and thus the timing of the future prepayments change, and thus timing of the future cash flows also changes.

To estimate the conditional hedge ratio, a structural model is usually required (as with model-based MBS valuation approaches). There are two drawbacks: First, there is no consensus regarding what is a reasonable specification of how the term structure moves through time, and how these movements relate to prepayment behavior. The model price is going to be closely related to these possibly ad-hoc assumptions, which may be reasonable or unreasonable.[31] Second, and more subtle, is the recognition that the parameter values themselves may often be "chosen" or estimated from a static viewpoint.[32] For instance, empirical prepayment models often reflect ad hoc prepayment rates on data sets housing and interest rate factors. But any of the well-documented MBS-hedging fiascoes would imply that the resulting regression coefficients, which present an average of the relation of the past, do not have the same link to the variable factors describing the current period. In other words, static in-sample regression estimated coefficients are not accurate estimators of future out-of-sample coefficients.

One method that has worked well in reducing the error between in-sample and out-of-sample estimators is the probability density estimation method.

The Multivariable Density Estimation Method

Multivariate density estimation (MDE) is a method for estimating the joint density of a set of variables. Given the joint and marginal densities of these variables, the corresponding distributions and conditional moments, such as the mean, can be calculated. The estimation relates the expected return on an MBS to the return on a T-note futures, conditional on relevant information available at any point in time. We have T observations, $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T$, where each $\mathbf{z}_t$ is an m-dimensional vector that might include the MBS and T-note futures returns, as well as several variables describing the state of the economy. One popular consistent measure of the joint density is the Parzen (fixed window width) density estimator:

$$\widehat{f}(\mathbf{z}^*) = \frac{1}{Th^m}\sum_{t=1}^{T} K\left(\frac{z^* - z_t}{h}\right)$$

where $K(\cdot)$ is called the kernel function (with the property that it integrates to unity) and is often chosen to be a density function, h is window or smoothing parameter (which helps determine how tight the kernel function is), and $\widehat{f}(z^*)$ is the estimate of the probability density at $\mathbf{z}^*$. The density at any point $\mathbf{z}^*$ is estimated as the average of densities centered at the actual data points $\mathbf{z}_t$. The further a data point is away from the estimation point, the less it contributes to the estimated density. Consequently, the estimate is highest near high concentrations of data points and lowest when observations are sparse.[33] A commonly used kernel is the multivariate normal density:

$$K(\mathbf{z}) = \frac{1}{(2\pi)^{m/2}} e^{-\frac{1}{2}\boldsymbol{z'z}}$$

Let $\mathbf{z}_t = (R^{mbs}_{t+1}, R^{TN}_{t+1}, \mathbf{x}_t)$, where R^{mbs}_{t+1} and R^{TN}_{t+1} are the one-period returns on the MBS and T-note futures from t and $t+1$, respectively, and $\mathbf{x}_t$ is an $(m-2)$-dimensional vector of factors known at time t. We can then obtain the conditional mean, $E\left[R^{mbs}_{t+1}|R^{TN}_{t+1}, \mathbf{x}_t\right]$—i.e., the expected MBS return given movements in the T-note return—conditional on the current economic state as described by $\mathbf{x}_t$. Specifically,

$$\begin{aligned} E\left[R^{mbs}_{t+1}|R^{TN}_{t+1}, \mathbf{x}_t\right] &= \int R^{mbs}_{t+1} \frac{f(R^{mbs}_{t+1}, R^{TN}_{t+1}, \mathbf{x}_t)}{f_1(R^{TN}_{t+1}, \mathbf{x}_t)} dR^{mbs} \\ &= \frac{\sum_{i=1}^{t} R^{mbs}_{t+1-i} K_1^{t-i}(\cdot,\cdot)}{\sum_{i=1}^{t} K_i^{t-i}(\cdot,\cdot)} \end{aligned} \tag{3.13}$$

where $K_1^{t-i}(\cdot,\cdot) = K_1((R^{TN}_{t+1-i} - R^{TN}_{t+1})/h^{TN}), (\mathbf{x}_{t-i} - \mathbf{x}_t)/h))$.

$K_1(\cdot,\cdot)$ is the marginal density, $\int K(z) dR^{mbs}$, which is also a multivariate normal density. The expected return in equation (3.13) is simply a weighted average of past returns where the weights depend on the levels of the conditioning variables relative to their levels in the past.

Given $E\left[R^{mbs}_{t+1}|R^{TN}_{t+1}, \mathbf{x}_t\right]$, a hedge ratio can be formed by estimating how much the return on the MBS changes as a function of changes in the T-note futures return, conditional on currently available information $\mathbf{x}_t$. That is

$$\frac{\partial E\left[R_{t+1}^{mbs}|R_{t+1}^{TN},\mathbf{x}_t\right]}{\partial R_{t+1}^{TN}} = \frac{\sum_{i=1}^{t} R_{t+1-i}^{mbs}\frac{\partial K_1^{t-i}(\cdot,\cdot)}{\partial R_{t+1}^{TN}}}{\sum_{i=1}^{t} K_1^{t-1}(\cdot,\cdot)} - \frac{\sum_{i=1}^{t} R_{t+1-i}^{mbs}K_1^{t-i}(\cdot,\cdot)\sum_{i=1}^{t}\frac{\partial K_1^{t-1}(\cdot,\cdot)}{\partial R_{t+1}^{TN}}}{\left[\sum_{i=1}^{t} K_1^{t-i}(\cdot,\cdot)\right]^2} \tag{3.14}$$

where

$$\frac{\partial K_1^{t-i}(\cdot,\cdot)}{\partial R_{t+1}^{TN}} = -\left[\frac{(R_{t+1-i}^{TN} - R_{t+1}^{TN})}{(h^{TN})^2}\right] K_1^{t-i}(\cdot,\cdot).$$

A couple of points can be made. First, equation (3.14) provides a formula for the hedge ratio between an investor's MBS position and T-note futures. For example, if $\frac{\partial E\left[R_{t+1}^{mbs}|R_{t+1}^{TN},\mathbf{x}_t\right]}{\partial R_{t+1}^{TN}}$ equals 0.5, then for every \$1 of an MBS held, the investor should short \$0.50 worth of T-note futures. Second, the hedge ratio will change dynamically, depending on the current economic state described by $\mathbf{x}_t$. For example, suppose $\mathbf{x}_t$ is an $m-2$ vector of term structure variables. As these variables change, whether they are the level, slope, or curvature of the term structure, the hedge ratio may change in response. Thus, the appropriate position in T-note futures will vary over time. Third, the hedge ratio is a function of the unknown return on the T-note futures. If the conditional relation between MBS returns and T-note futures returns is always linear, then the same hedge ratio will be appropriate, regardless of how T-note futures move. If the relation is not linear, then the investor must decide what type of T-note moves to hedge. For example, the investor might want to form the MBS hedge in the neighborhood of the conditional mean of the T-note futures return because many of the potential T-note futures will lie in that region. On the other hand, it may be the case that the investor is concerned about the tails of the distribution T-note futures returns, and thus adjusts the hedge ratio to take account of potential extreme moves in interest rates and T-note futures. Fourth, the hedge ratio is horizon specific. In contrast to the instantaneous hedge ratio, the method's implied hedge ratio directly reflects the distribution of MBS returns over the relevant horizon. Thus, different hedge ratios may be appropriate for daily, weekly, or monthly horizons.

The static OLS regression coefficient, or hedge ratio, is given by

$$\beta = \frac{\sum_{i=1}^{t} R_{t+1-i}^{mbs}R_{t+1-i}^{TN} - T\mu_{mbs}\mu_{TN}}{\sum_{i=1}^{t}(R_{t+1-i}^{TN} - \mu_{TN})^2} \tag{3.15}$$

where $\mu_{mbs} = \frac{1}{T}\sum_{i=1}^{t} R_{t+1-i}^{mbs}$ and $\mu_{TN} = \frac{1}{T}\sum_{i=1}^{t} R_{t+1-i}^{TN}$.

In contrast, the dynamic hedging method explicitly takes into account the current economic state. Equation (3.14) can be rewritten as

$$\frac{\partial E\left[R_{t+1}^{mbs}|R_{t+1}^{TN},\mathbf{x}_t\right]}{\partial R_{t+1}^{TN}} = \sum_{i=1}^{t} R_{t+1-i}^{mbs}\left[\frac{(R_{t+1-i}^{TN}-R_{t+1}^{TN})}{(h^{TN})^2}\right] w_i(t) - \left[\sum_{i=1}^{t} R_{t+1-i}^{mbs} w_i(t)\right] \cdot \left[\sum_{i=1}^{t}\left(\frac{(R_{t+1-i}^{TN}-R_{t+1}^{TN})}{(h^{TN})^2}\right) w_i(t)\right] \tag{3.16}$$

where $w_i(t) = \frac{K_1^{t-i}(\cdot,\cdot)}{\sum_{i=1}^{t} K_1^{t-i}(\cdot,\cdot)}$.

The hedge ratio in (3.16) is constructed by taking past pairs of MBS and T-note futures returns, and then differentially weighting these pairs' co-movements by determining how "close" $(R_{t+1-i}^{TN}, \mathbf{x}_{t-i})$ pairs are to a chosen value of R_{t+1}^{TN} and current information $\mathbf{x}_t$. The dynamic hedge ratio is similar in spirit to a regression hedge, except that the weights are no longer constant, but instead depend on current information. $w_i(t)$ puts little weight on the observation pair $(R_{t+1-i}^{mbs}, R_{t+1-i}^{TN})$ if the current information $\mathbf{x}_t$ is not close to $\mathbf{x}_{t-i}$ in a distributional sense. The hedge ratio adjusts to current economic conditions. For example, if interest rates are currently high, but the term structure is inverted, then more weight will be given to past co-movements between MBS and T-note futures in that type of interest rate environment.

Boudoukh, Richardson, Stanton, and Whitelaw (1995) apply the method to weekly 30-year fixed-rate GNMA MBS (with 8%, 9%, and 10% coupons) and T-note futures data over the period January 1987 to May 1994. The GNMA prices represent dealer-quoted prices on $X\%$ coupon-bearing GNMAs traded for forward delivery on a *to be announced* (TBA) basis.[34] Performing an out-of-sample analysis, their research shows that the dynamic hedging method performs considerably better than the static regression method. For instance, in hedging weekly returns on 10% GNMA, the dynamic method reduces the volatility of the GNMA return from 41 to 24 basis points, whereas a static method manages only 29 basis points of residual volatility. Furthermore, only 1 basis point of the volatility of the dynamically hedged return can be attributed to risk associated with U.S. Treasuries in contrast to 14 basis points of interest rate risk in the statically hedged return.

The results of Boudoukh, Richardson, Stanton, and Whitelaw (1995) shown in Table 3.26 compares the mean, volatility, and autocorrelation of unhedged returns on GNMA TBAs and hedged returns using two different approaches. The approaches involve hedging GNMAs with T-note futures, resulting in the hedged return, $R_{t+1}^{mbs} - \beta_{t+} R_{t+1}^{TN}$, where R_{t+1}^{mbs} and R_{t+1}^{TN} are the out-of-sample returns on GNMAs and T-note futures respectively, and the hedge ratio, is estimated using the prior 150 weeks of data in one of two ways: (1) a linear hedge based on a regression of past R_{t+1}^{mbs} on R_{t+1}^{TN}, and (2) a MDE hedge using the distribution of R_{t+1}^{mbs} and R_{t+1}^{TN}, conditional on the 10-year yield at time t. The estimation is performed on a rolling basis and covers the out-of-sample period, December 1989 to May 1994. Results are reported for both weekly and overlapping monthly returns.

Table 3.26

	GNMA 8		GNMA 9		GNMA 10	
	1 wk.	4 wks.	1 wk.	4 wks.	1 wk.	4 wks.
Unhedged						
Mean (%)	.078	.326	.077	.316	.069	.286
Vol. (%)	.685	1.364	.531	.999	.414	.746
Autocorr.	.011	.138	-.019	.109	-.045	.082
Linear Hedge						
Mean (%)	.007	.041	.017	.086	.024	.126
Vol. (%)	.318	.599	.309	.573	.286	.524
Autocorr.	.015	-.107	.012	.021	.060	.039
MDE Hedge						
Mean (%)	.006	.123	.020	.178	.027	.189
Vol. (%)	.285	.583	.245	.472	.242	.440
Autocorr.	.011	-.096	.016	-.083	.036	-.085

Source: Boudoukh, Richardson, Stanton, and Whitelaw (1995)

Figure 3.10 shows hedge ratios for hedging weekly 10% (top) and 8% (bottom) GNMA returns using the 10-year T-note futures. Hedge ratios are estimated on a 150-week rolling basis using both a linear regression and MDE. The MDE hedge ratios condition on the level of the 10-year T-note yield.

Figure 3.11 shows the expected weekly return on a 10% (top) and an 8% (bottom) GNMA as a function of the contemporaneous 10-year T-note futures return, conditional on three different levels of the 10-year T-note yield. The relation is estimated using MDE over the period January 1987 to May 1994. Returns are in percent per week.

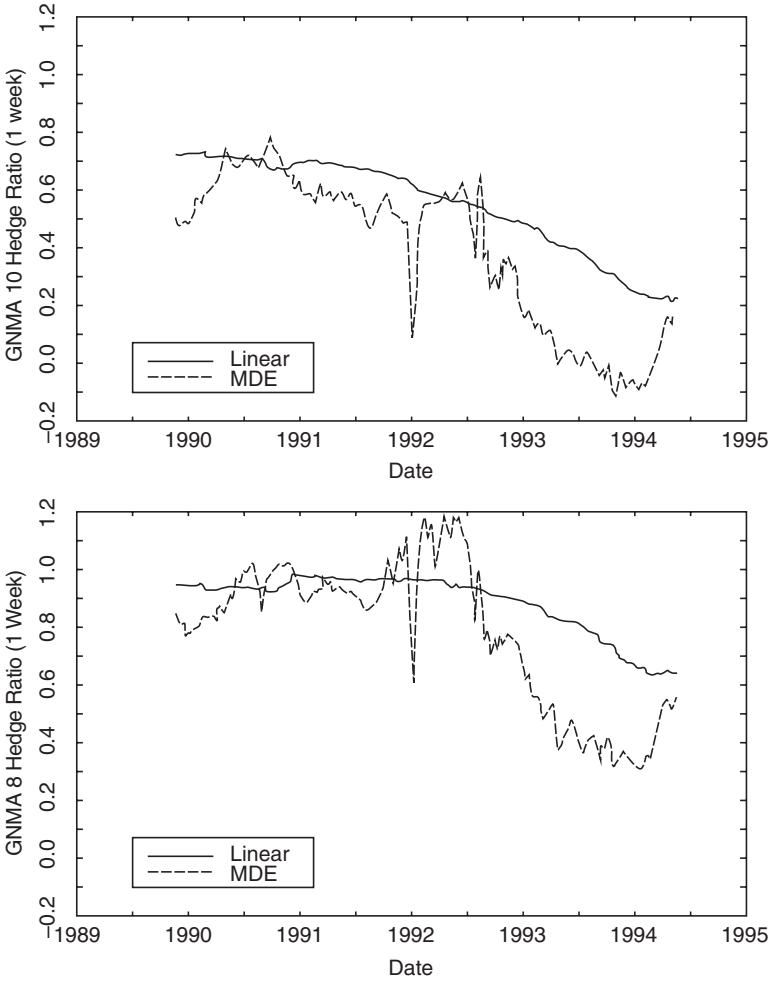

Figure 3.10 Hedge ratios for hedging weekly 10% (top) and 8% (bottom) GNMA returns using the 10-year T-note futures. *Source: Boudoukh, Richardson, Stanton, and Whitelaw (1995).*

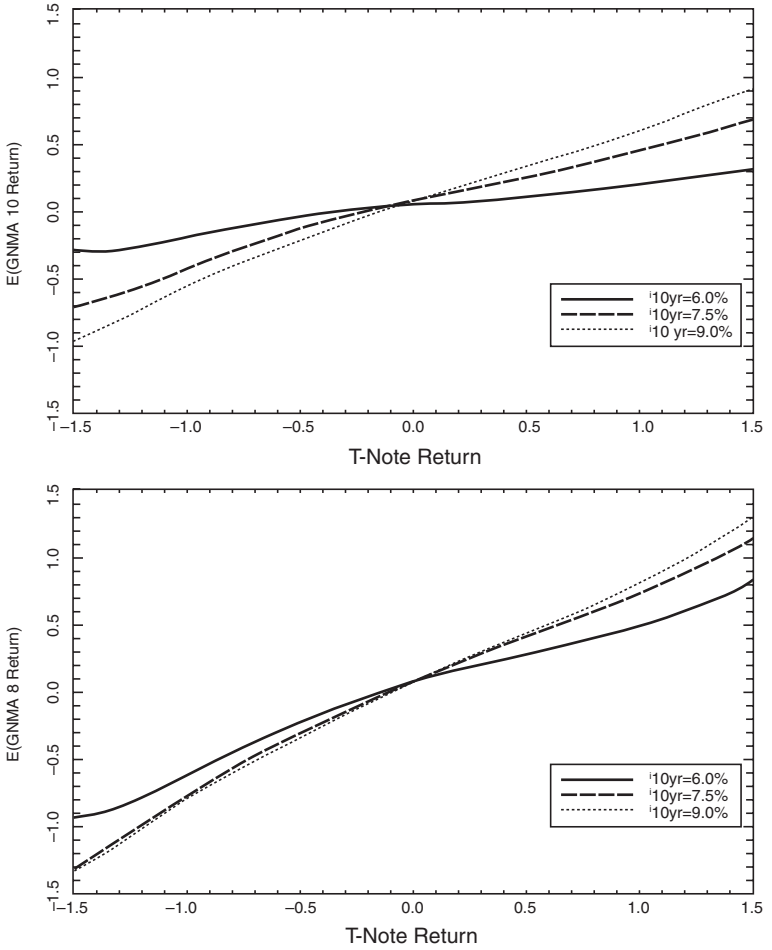

Figure 3.11 Expected weekly return on a 10% (top) and an 8% (bottom) GNMA as a function of the contemporaneous 10-year T-note futures return, conditional on three different levels of the 10-year T-note yield. *Source: Boudoukh, Richardson, Stanton, and Whitelaw (1995).*

ENDNOTES

1. Throughout this chapter, we follow the direct work of Dr. Stafford Johnson, Professor of Finance at Xavier University, at *http://www.academ.xu.edu/johnson/*.
2. Obazee, P. (2002), pp. 338–339. See "Understanding the Building Blocks for OAS Models" in *Interest Rate, Term Structure, and Valuation Modeling*. Edited by Frank J. Fabozzi, Wiley & Sons.
3. Id., pp. 338–339.
4. Id., pp. 338–339.
5. In practice, one would use a more sophisticated term structure model than the binomial model such as the Hull-White, Black-Derman-Toy, Black-Karasinski, or Cox-Ingersoll-Ross interest rate models.
6. We give the C++ code later in the chapter.
7. Hull, J. (1996), pg. 391.
8. Fabozzi, F., Richard, S., and Horwitz, D. (2002), pp. 445. "Monte Carlo Simulation/OAS Approach to Valuing Residential Real Estate-Backed Securities" in *Interest Rate, Term Structure, and Valuation Modeling*, Wiley & Sons.
9. Id., pp. 445–446.
10. Id., pp. 445–446.
11. Id., pg. 446.
12. Id., pg. 453.
13. Obazee, P. (2002), pp. 315–344. See "Understanding the Building Blocks for OAS Models" in *Interest Rate, Term Structure, and Valuation Modeling*. Edited by Frank J. Fabozzi, Wiley & Sons.
14. Id., pg. 317.
15. Id., pg. 318.
16. Id., pg. 319.
17. Bandic, I., pg. 11.
18. The refinancing incentive and seasoning factor were previously discussed, but are now given analytical formula specifications based on Richard and Roll's (1989) empirical observations.
19. Formulas from Davidson/Herskovitz (1996).
20. Monthly parameters were taken from Figure 3 in Richard Roll (1989).
21. Fabozzi, F., Richard, S., and Horwitz, D. (2002), pg. 459. "Monte Carlo Simulation/OAS Approach to Valuing Residential Real Estate-Backed Securities" in *Interest Rate, Term Structure, and Valuation Modeling*, Wiley & Sons.
22. Id., pg. 454.
23. Fabozzi, F., Richard, S., and Horwitz, D. (2002), pg. 454.
24. Id., pg. 455.
25. The one-factor model has the capacity to describe the burnout effect of prepayment by embedding heterogeneity of prepayment behavior into the MBS valuation as a function of mortgage rates. In contrast, the three-factor model is based on discrete-time, no-arbitrage pricing theory, making an association between prepayment behavior and cash flow patterns where prepayment behavior is due to refinancing(caused by changes in interest rates) and

rising housing prices by incentive response functions. (See Kariya and Kabayashi, 2000, and Kariya, Ushiyama, and Pliska, 2002.)

26. Boudoukh, Richardson, Stanton, and Whitelaw (1995), pg. 1.
27. Id., pg. 1.
28. Id., pg. 1.
29. For a discussion of some of the problems associated with static hedges, see, for example, Breeden (1991) and Breeden and Giarla (1992). With respect to linear regression hedges in particular, Batlin (1987) discusses the effect of the prepayment option on the hedge ratio between MBSs and T-note futures.
30. Davidson and Herkowitz (1992) provide an analysis of the various theoretical methodologies for valuing MBSs in practice. The advantages and disadvantages of each approach are discussed in detail. With respect to the particular issue of hedging MBSs, Roberts (1987) gives an analysis, focusing primarily on model-based approaches to MBS valuation.
31. Id.
32. Id., pg. 2.
33. Id., pg. 3.
34. The TBA market is most commonly employed by mortgage originators who have a given set of mortgages that have not yet been pooled. However, trades can also involve existing pools, on an unspecified basis. This means that, at the time of the agreed-upon transaction, the characteristics of the mortgage pool to be delivered (e.g., the age of the pool, its prepayment history, and so on) are at the discretion of the dealer. Nonetheless, as long as new mortgages with the required coupon are being originated, these pools are likely to be delivered because seasoned pools are more valuable in the interest rate environment that characterizes a sample period. Thus, GNMA TBAs are best viewed as forward contracts on generic, newly issued, securities.