Methods for Estimating Term Structure of Interest Rates

Iskander Karibzhanov

Abstract This paper compares different interpolation algorithms for constructing yield curves: cubic splines, linear and quadratic programming, binomial functions and parametric methods. Results confirm that the constrained Schaefer method produces smoother and more stable curves than does unconstrained Schafer's approximation, cubic splines, or discrete approximation. Adding monotonicity constraints in Schaefer method does not increase computational burden as it does in the discrete approximation. My experiments found that adding monotonicity constraints on cubic spline method is unfeasible. I also experimented with three well know parsimonious functional approximations of the term structure: the Nelson-Siegel, Svensson and Bliss exponential functions. Accompanying MATLAB software package is available from the author's website and illustrates practical application using U.S. treasury bond market data set.

Keywords term structure estimation · yield curve · fixed income · bond pricing

1 Introduction

There are several commonly used techniques for estimation of the term structure of interest rates: regression analysis with cubic splines by Litzenberger and Rolfo (1984), binomial functions by Schaefer (1981), and parameterized methods utilizing parsimonious exponential functions by Nelson and Siegel (1987), Söderlind and Svensson (1997) and Bliss (1997).

In linear programming and regression methods for term structure estimation, the estimator is a solution to an optimization problem. In linear programming, the estimator is the maximizer of the present value of a prescribed sequence of cash flows, subject to the present value of each bond being less than or equal to its price. In regression method, the estimator is the minimizer of the sum of squared deviations of the bond prices from their present values.

I. Karibzhanov
Bank of Canada, 234 Wellington Street, Ottawa, Ontario, K1A 0G9, Canada
Tel.: +1-613-782-8623, E-mail: kais@bankofcanada.ca, URL: http://karibzhanov.com

Section 2 explains the methodology by presenting one discrete and four continuous approximation methods of term structure estimation. Section 3 presents the results of my calculations and compares the estimated term structures across various types of bonds. I find that Schaefer method produces more stable and smoother curves than other methods. Computational Appendix A discusses implementation issues and serves as documentation for MATLAB codes. It describes how to import bond data from text files, construct cash flows matrix, compute tenor periods and impose monotonicity constraints on quadratic optimization.

2 Methodology

2.1 Discrete Approximation

The process of using a discrete approximation to estimate the term structure from a given sample of M bonds can be broken down into four steps:

1. Construct $N \times 1$ vector $\mathbf{t}$ containing unique and monotonically increasing dates at which any coupon or principal payment is made,
2. Construct $M \times N$ matrix $\mathbf{A}$ in which each row represents the cash flow structure of a particular bond mapped to corresponding dates in vector $\mathbf{t}$,
3. Construct $M \times 1$ vector $\mathbf{p}$ containing the bonds' cash prices by adding accrued interests to quoted prices,
4. Solve one of the three problems:
   - solve the *least squares problem* of finding the $N \times 1$ vector of discount factors $\mathbf{d}$, which would minimize the norm $\|\mathbf{Ad} - \mathbf{p}\|^2$, *or*
   - add monotonicity constraints on the discount factors and solve *quadratic programming* problem to find the closest fit to minimize the norm $\|\mathbf{Ad} - \mathbf{p}\|^2$, *or*
   - add monotonicity constraints on the discount factors and solve a *linear programming* problem by maximizing the present value of future cash flows subject to the present value of each bond being less than or equal to its actual price. Note that this linear programming problem is much easier to solve and more accurate than quadratic programming or least squares problems.

2.2 Continuous Approximation

Let's first give definition to discount function, yield curve and forward interest rate curve. Discount function, denoted as $d(t)$, is equal to present value of discount (zero-coupon) risk-free bond paying one dollar at time t. Continuously compounding interest rate on this type of bond is called spot rate and is denoted as $r(t)$. By changing maturity date t, one can obtain a plot of spot interest rates or a (spot) yield curve. Let $f(t)$ denote instantaneous forward rate, i.e. rate on forward contract with maturity date equal to settlement date. Then, we have the following relationships between $d(t)$, $r(t)$ and $f(t)$:

$$d(t) = \exp(-r(t)t) = \exp\left(-\int_{s=0}^{s=t} f(s)\,ds\right) \tag{1}$$

$$r(t) = -\frac{\ln(d(t))}{t} = \frac{1}{t}\int_{s=0}^{s=t} f(s)\,ds \tag{2}$$

$$f(t) = -\frac{d(t)'}{d(t)} = r(t) + r(t)'t \tag{3}$$

The main drawback of the discrete approximation is that it estimates the discount rates only at the particular time values $t_1, t_2, \ldots, t_N$. Continuous approximations avoid this by assuming that the discount rate takes a continuous functional form such as:

$$d(t) \approx \sum_{k=0}^{K} x_k b_k(t), \tag{4}$$

where $b_k(t)$ — are specified component functions of t.

Then the estimation problem consists of computing the $K+1$ parameters x_k, such that $d(t)$ is consistent with market prices. This can be done by minimizing $\|\mathbf{ABx} - \mathbf{p}\|^2$ with respect to $\mathbf{x}$, where

$\mathbf{A}$ — $M \times N$ matrix of cash flows described above,

$\mathbf{B}$ — functional $N \times (K+1)$ matrix, whose i–th row is given by

$$b(t_i) = [b_0(t_i), b_1(t_i), \ldots, b_K(t_i)], \tag{5}$$

$\mathbf{x}$ — decision vector $[x_0, x_1, \ldots, x_K]'$.

2.2.1 Cubic Splines Regression Model by Litzenberger and Rolfo

Among the earlier methods of estimating term structure are those of Litzenberg and Rolfo. These authors utilize regression methodology to find the present value factors, or, equivalently, the term structure of spot interest rates, that best "explain" the observed prices of coupon bonds. Using multivariate linear regression analysis, the difference between a bond's price and its present value is minimized. This regression analysis incorporates smoothing of the term structure with cubic splines.

In the cubic spline method, $\{\tau_k\}_{k=0}^{m} \in [0, T]$ denotes a set of knot points over the interval in which coupon payments are made for which $0 \equiv \tau_0 < \tau_1 < \cdots < \tau_m \equiv T$. The knots are placed such that an equal number of payment dates falls into each subinterval. The value for m is taken to be the integer closest to the square root of the total number of bonds in the marketplace ($m \approx \sqrt{M}$). The value of r may be interpreted as the sample size but should not be less than 10. The cubic splines approximate $d(t)$ by

$$d(t) \approx 1 + x_1 t + x_2 t^2 + x_3 t^3 + \sum_{k=1}^{m} x_{k+3}(t - \tau_k)^3 \mathrm{I}_{t \geq \tau_k}, \tag{6}$$

where $\mathrm{I}_{t \geq \tau_k} = 1$ when $t \geq \tau_k$ and $\mathrm{I}_{t \geq \tau_k} = 0$ when $t < \tau_k$.

For each realized payment date t_i, let $b(t_i)$ denote the vector

$$b(t_i) = \left[1, t_i, t_i^2, t_i^3, (t_i-\tau_1)^3 \mathrm{I}_{t_i \geq \tau_1}, (t_i-\tau_2)^3 \mathrm{I}_{t_i \geq \tau_2}, \ldots, (t_i-\tau_m)^3 \mathrm{I}_{t_i \geq \tau_m}\right], \quad (7)$$

and $\mathbf{B}$ denotes the $N \times (4+m)$ matrix whose i–th row is given by $b(t_i)$. The estimation problem is then to minimize the norm $\|(\mathbf{AB})\mathbf{x} - \mathbf{p}\|^2$ for $\mathbf{x} = [1, x_1, x_2, \ldots, x_{3+m}]'$.

The cubic splines are designed to be continuous and have both continuous first and second derivatives. Therefore we can derive the following forward rate curve from discount function of cubic spline method:

$$f(t) = -\frac{d(t)'}{d(t)} \approx -\frac{x_1 + 2x_2 t + 3x_3 t^2 + 3\sum_{k=1}^{m} x_{k+3}(t-\tau_k)^2 \mathrm{I}_{t \geq \tau_k}}{1 + x_1 t + x_2 t^2 + x_3 t^3 + \sum_{k=1}^{m} x_{k+3}(t-\tau_k)^3 \mathrm{I}_{t \geq \tau_k}} \quad (8)$$

However, cubic spline is not guaranteed to be a monotonically decreasing function. To ensure monotonicity, I estimated the term structure using Schaefer method.

2.2.2 Schaefer Method

In order to implement Schaefer method, one must normalize time vector $\mathbf{t}$, i.e. we must scale each payment date by the longest one so that time is measured on the interval $[0,1]$. The component functions $b_k(t)$ are defined as follows:

$$b_0(t) \equiv 1, \quad (9)$$

and for each $k = 1, 2, \ldots, K$,

$$b_k(t) = -\int_0^t u^{k-1}(1-u)^{K-k}\,du = \sum_{j=0}^{K-k} (-1)^{j+1} C_j^{K-k} \left(\frac{t^{k+j}}{k+j}\right) \quad (10)$$

where $C_n^k = \frac{n!}{k!(n-k)!}$ and K is usually taken to be 25.

Each x_k is constrained to be non-negative, and each $b_k(t)$ is a monotonically decreasing nonpositive function on the interval $[0,1]$. Therefore, Schaefer method ensures that $d(t)$ is a monotonically decreasing function. To ensure that $d(t)$ are non-negative, one also adds the constraint

$$\sum_{k=0}^{K} x_k b_k(1) \geq 0. \quad (11)$$

and the parameters are estimated as they were before.

2.2.3 Nelson-Siegel Exponential Functions Method

Rather than explicitly modeling the term structure, one may want to approximate it by a flexible functional form. There are several parsimonious exponential models, proposed by Nelson and Siegel (1987), Söderlind and Svensson (1997) and Bliss (1997).

The Nelson-Siegel approximation is derived from the assumption that the spot rates follow a second-order differential equation and that forward rates, which are

the predicted spot rates, are the solution to this differential equation with equal roots. Let's assume that the equation for instantaneous forward rate is given in the following form:

$$f(t) = \beta_0 + \beta_1 \exp\left(-\frac{t}{\lambda}\right) + \beta_2 \frac{t}{\lambda} \exp\left(-\frac{t}{\lambda}\right) \tag{12}$$

Then the yield curve can be computed using formula 2 to obtain:

$$r(t) = \beta_0 + (\beta_1 + \beta_2)\left[\frac{1 - \exp\left(-\frac{t}{\lambda}\right)}{\frac{t}{\lambda}}\right] - \beta_2 \exp\left(-\frac{t}{\lambda}\right). \tag{13}$$

Söderlind and Svensson (1997) improved the Nelson-Siegel model by formulating forward rates and spot rates as follows:

$$f(t) = \beta_0 + \beta_1 \exp\left(-\frac{t}{\lambda_1}\right) + \beta_2 \frac{t}{\lambda_1} \exp\left(-\frac{t}{\lambda_1}\right) + \beta_3 \frac{t}{\lambda_2} \exp\left(-\frac{t}{\lambda_2}\right) \tag{14}$$

$$r(t) = \beta_0 + (\beta_1 + \beta_2)\left[\frac{1 - \exp\left(-\frac{t}{\lambda_1}\right)}{\frac{t}{\lambda_1}}\right] - \beta_2 \exp\left(-\frac{t}{\lambda_1}\right) + \beta_3\left[\frac{1 - \exp\left(-\frac{t}{\lambda_2}\right)}{\frac{t}{\lambda_2}} - \exp\left(-\frac{t}{\lambda_2}\right)\right] \tag{15}$$

The Nelson-Siegel method also took further development by Bliss (1997). His improved approximation is given as:

$$f(t) = \beta_0 + \beta_1 \exp\left(-\frac{t}{\lambda_1}\right) + \beta_2 \frac{t}{\lambda_2} \exp\left(-\frac{t}{\lambda_2}\right) \tag{16}$$

$$r(t) = \beta_0 + \beta_1\left[\frac{1 - \exp\left(-\frac{t}{\lambda_1}\right)}{\frac{t}{\lambda_1}}\right] + \beta_2\left[\frac{1 - \exp\left(-\frac{t}{\lambda_2}\right)}{\frac{t}{\lambda_2}} - \exp\left(-\frac{t}{\lambda_2}\right)\right] \tag{17}$$

In each of the above models, parameters $\beta_0, \lambda, \lambda_1$ and λ_2 must be positive. The parameter $\frac{1}{\lambda}$ governs the exponential decay rate; small values of $\frac{1}{\lambda}$ produce slow decay and can better fit the curve at long maturities, while large values of $\frac{1}{\lambda}$ produce fast decay and can better fit the curve at short maturities. We can interpret $\beta_0, \beta_1, \beta_2$ as three latent factors. The loading on β_0 is a constant that does not decay to zero in the limit; thus, it may be viewed as a long-term factor. The loading on β_1 is $\left[1 - \exp\left(-\frac{t}{\lambda_1}\right)\right]\frac{\lambda_1}{t}$, which starts at 1 but decays quickly and monotonically to 0; hence, β_1 may be viewed as a short term factor. The loading on β_2 is $\left[1 - \exp\left(-\frac{t}{\lambda_2}\right)\right]\frac{\lambda_2}{t} - \exp\left(-\frac{t}{\lambda_2}\right)$, which starts at 0 (and is thus not short-term), increases, and then decays to zero (and thus is not long term); hence, β_2 can be interpreted as a medium term factor.

β_0, β_1 and β_2 can also be interpreted in terms of the aspect of the curve that they govern: level, slope, and curvature. The long-term factor β_0 governs the yield curve level. In particular, $r(\infty) = \beta_0$. Alternatively, note that an increase in β_0 augments all

yields equally, as the loading is identical at all maturities. The short-term factor β_1 is equal to the yield curve slope, $r(\infty) - r(0)$. Note that an increase in β_1 augments short yields more than long yields because the short rates load on β_1 more heavily, thereby changing the slope of the yield curve. Finally, β_2 is closely related to yield curvature: an increase in β_2 will have very little effect on very short or very long yields, which load minimally on it, but will increase medium-term yields, which load more heavily on it, thereby increasing yield curve curvature.

Let θ denote the set of five parameters discussed above, i.e.

$$\theta = \{\beta_0, \beta_1, \beta_2, \lambda_1, \lambda_2\}. \tag{18}$$

The parameters can be estimated using nonlinear least-squares data fitting by the Gauss-Newton method, implemented in `nlinfit` function from MATLAB Statistics Toolbox, i.e.

$$\hat{\theta} = \arg\min \sum_{i=1}^{N} \varepsilon_i^2, \tag{19}$$

where ε_i is the difference between the actual market yields and theoretical fitted yields on i–th bond at time $t = 0$ in both methods.

3 Results

3.1 Comparison of Approximation Methods

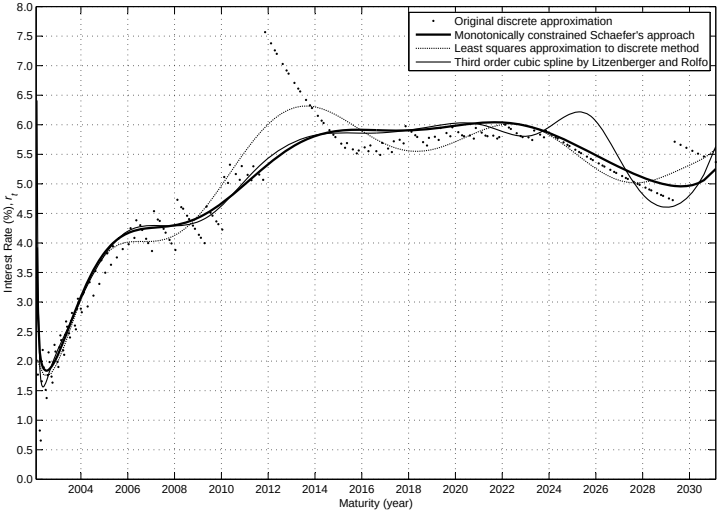
In this paper I use five methods to estimate the term structure from different kinds of bonds - one discrete approximation and four continuous approximations. For the discrete approximation, the objective function is given in terms of discount factors $d(t)$ on discrete dates t. I simply find the vector $\mathbf{d}$ such that the sum of the squared errors $\|\mathbf{Ad} - \mathbf{p}\|^2$ is minimized. The model, therefore, is easy to formulate. However, the quality of the solutions is not likely to be satisfying since discrete approximations rarely give smoothed curves.

Let's now consider the first two continuous methods. The first one is the cubic spline method, which defines a new set of variables, x_i. This method tries to fit the values of each x_i such that the sum of the squared errors $\|(\mathbf{AB})\mathbf{x} - \mathbf{p}\|^2$ is minimized. To formulate this model, we have to construct a new matrix, $\mathbf{B}$, where $\mathbf{d}$ is estimated to equal to $\mathbf{Bx}$. This method produces better solutions than those from discrete method.

In the second continuous approximation, Schaefer method, we also have to construct a new matrix, $\mathbf{B}$, and a set of variables $\mathbf{x}$. The objective of this method is the same as the former two; that is, to minimize the sum of the squared errors. However, in this method, unlike in the former two, time is scaled such that it is measured on the interval $[0, 1]$.

I found that the curves produced using Schaefer method almost always look smoother than those of the other two methods. This method also gives me the monotonically decreasing function of $d(t)$ by simply setting the lower bound of the variables to zero. In practice, it takes a short time to solve for the optimal solution while

Fig. 1 Comparison of discrete, least squares, Schaefer and cubic approximations



This figure shows how noise in discrete method can be smoothed by using least squares approximation from MATLAB splines toolbox. Cubic splines produce more volatile solution compared to the Schaefer method.

the discrete approximation with a set of monotonic constraints takes a much longer time to solve for a set of solutions.

Figure 1 illustrates the treasury yield curves obtained using each method. I found that the curve from Schaefer method produces the smoothest curve while the discrete approximation produced the worst curve. The continuous methods also do better in avoiding the unreasonable fluctuations in the curve.

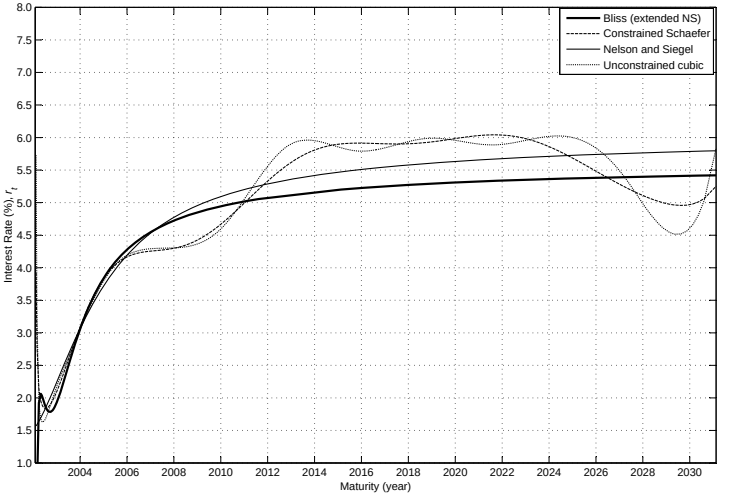
Figure 2 compares Nelson-Siegel, Bliss, cubic and Schaefer approximations. The exponential function approximation proposed by Nelson and Siegel and its further development by Robert Bliss resemble the embedded exponential shape. As we expected, the exponential form is much less volatile compared to cubic splines and Schaefer method and thus is most preferred in practice.

3.2 Estimated Term Structure Across Various Bond Types

My estimates for the term structure using a discrete approximation, cubic splines and Schaefer method were calculated using data from treasury coupon securities, zero coupon bonds (STRIPS), AA government bonds, and AAA corporate bonds. Figure 3 shows the term structure as estimated by Schaefer method with the constraints $d(t) \geq 0$ (non-negativity) and $d_1 \geq d_2 \geq \cdots \geq d_N$ (monotonicity) for each of the aforementioned types of bonds.

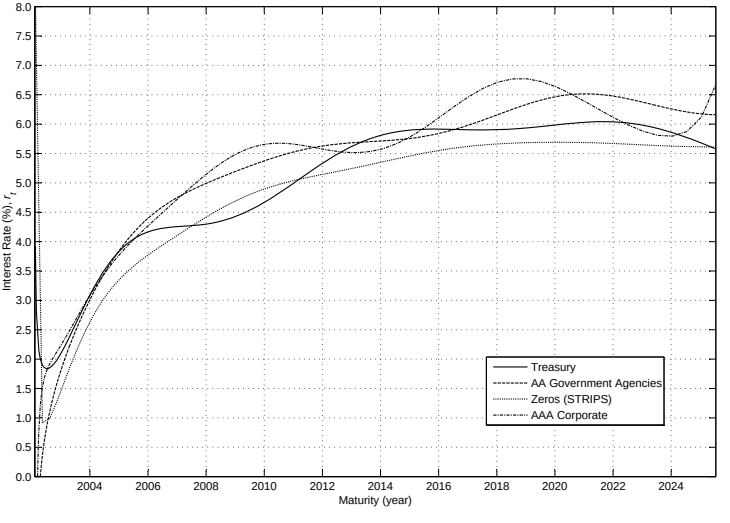
As 'zeroes' only pay the principals at maturities and do not make periodic coupon payments, they produce the 'smoothest' term structure curve as can be seen by the lack of oscillations. Only the short rates between maturity dates, when the principal is paid to the investor, need be considered when estimating the term structure with

Fig. 2 Comparison of Bliss, Nelson-Siegel, Schaefer and cubic approximations



This figure shows that the smooth exponential functional approximation proposed by Nelson-Siegel and Bliss is the preferred method to estimate yield curves compared with volatile cubic splines and Schaefer approximations.

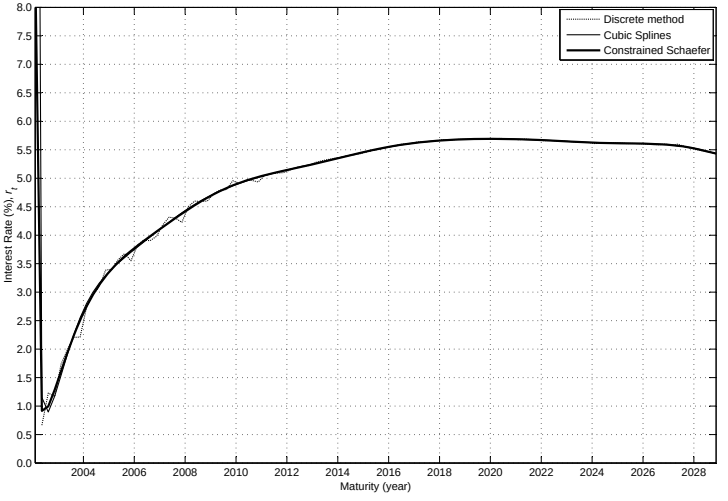
Fig. 3 Term structure for various types of bond by constrained Schaefer method



zeroes. Note that a smooth yield curve is always produced when the term structure is estimated from STRIPS regardless of whether a discrete or continuous approximation is used (figure 4).

Furthermore, zeroes and treasuries produce the lowest yield curves - i.e., they predict lower yields to maturity compared to the estimates made using other bonds. This is because zeroes and treasuries are backed by the full faith and credit of the

Fig. 4 Term structure of zero-coupon bonds (STRIPS)



In case of STRIPS, all of the considered methods produce very close solutions due to the fact that zero coupon bonds do not bear intermediate coupons payments that create volatility in estimating term structure.

U.S. government and thus have no credit risk; hence, they have lower risk premiums (smaller yields) than other types of bonds.

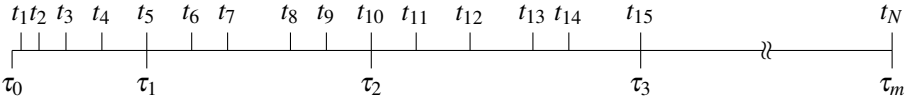
Bonds issued by government sponsored enterprises (GSEs) such as the Federal Home Loan Mortgage Corporation and the Federal National Mortgage Association are privately owned and publicly chartered entities; they carry slightly more credit risk than do treasuries.

Corporate bonds carry the most credit risk since the ability of a firm to make timely principal and coupon payments depends on how successful the firm is, which varies from fiscal year to fiscal year. These varying susceptibilities to credit risk are in accordance with figure 3, in which the corporate bond yield curve tends to be higher than the AA government bond curve, which, in turn, is higher than the yield curve estimated from zeroes.

Treasuries, GSE securities, and corporate bonds are all equally susceptible to interest rate risk, inflation risk, and reinvestment risk. Although the U.S. government backs both zeroes and treasuries, since zeroes do not pay coupons, they are not susceptible to reinvestment risk, which may explain why they tend to predict lower yields to maturity than do treasury coupon securities.

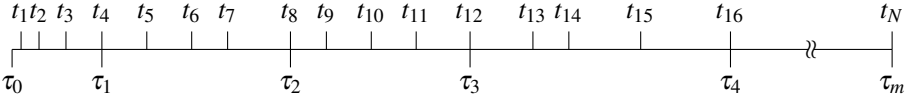
3.3 Effect of Including More Terms in the Continuous Approximation

For continuous approximations, I used the cubic splines and Schaefer method. In the cubic spline method, I define the knot points $\tau_0, \tau_1, \ldots, \tau_m$ such that an equal number of payment dates falls into each subinterval:



We know that $d(t) \approx 1 + x_1 t + x_2 t^2 + x_3 t^3 + \sum_{k=1}^{m} x_{k+3}(t-\tau_k)^3 \mathrm{I}_{t \geq \tau_k}$, where $\mathrm{I}_{t \geq \tau_k} = 1$ when $t \geq \tau_k$ and $\mathrm{I}_{t \geq \tau_k} = 0$ when $t < \tau_k$. If we closely look at this condition, we will see that each $d(t)$ will be made up of different numbers of variables x_i, i.e. in the example above, $d(t_1), d(t_2), d(t_3), d(t_4), d(t_5)$ are each the linear combination of three variables x_1, x_2 and x_3, while each of $d(t_6), d(t_7), d(t_8), d(t_9)$ and $d(t_{10})$ is the linear combination of four variables x_1, x_2, x_3 and x_4, and each of $d(t_{11}), d(t_{12}), d(t_{13}), d(t_{14}), d(t_{15})$ is the linear combination of five variables x_1, x_2, x_3, x_4 and x_5, and so on.

In each subinterval, the values of $d(t)$ will not be much different since they are estimated from the same set of variables. Nevertheless, when we add more terms in the approximation, i.e. when we increase m, the number of payment dates in each subinterval will decrease. This, therefore, leads to the decreasing of smoothness of $d(t)$.

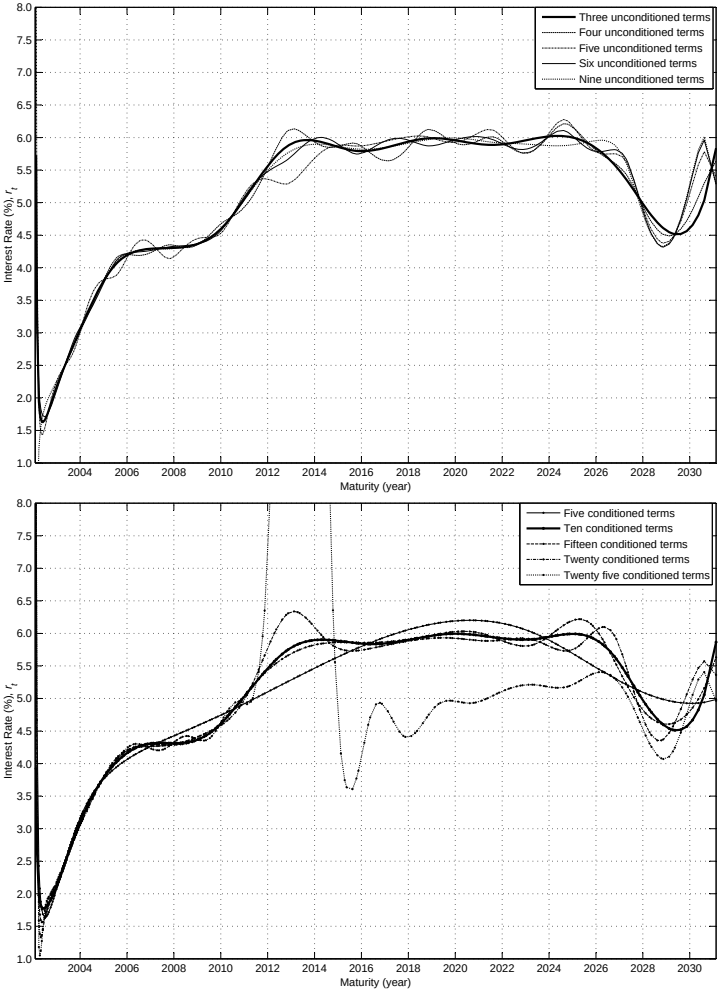


Top panel of figure 5 illustrates the effect of including more conditioned terms in the cubic spline approximation 6. I found that the best solution is obtained if I set $m = 10$ as it was recommended integer closest to the square root of the number of bonds (135 for treasury securities). When I increased the value of m to 20, the curve swung slightly higher than it did when when m was equal to 15. For m equal to 25, the interest rate curve deviated wildly. I found that the curve did not oscillate so frequently if m was decreased to 10 or 5. The number of terms can be interpreted as our sample size. Including more terms in cubic spline approximation increases volatility. This is evident in case of 25 terms. However, the allowable number of terms should be at least 10. Otherwise, the term structure would be overly smoothed (as in the case of 5 terms).

I also considered the effect of including more unconditioned terms, i.e. those of the from $x_i t^i$ in cubic spline approximation 6. Bottom panel of figure 5 shows term structures as estimated with treasuries using the cubic splines with 3, 4, 5, 6, and 9 unconditioned terms. Including more than three unconditioned terms in cubic spline approximation results in higher volatility and hence not recommended. When 4, 5, or 6 terms are used, the yield curve estimation oscillated noticeably. The 3–term curve and 9–term curve seemed to produce the best fit.

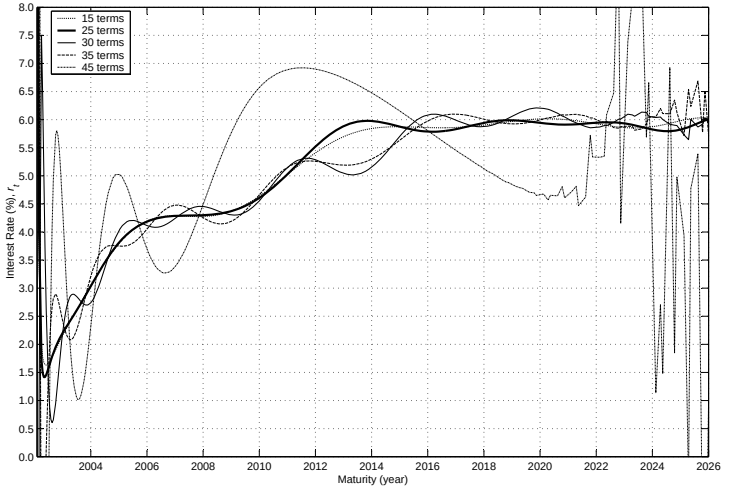
In Schaefer method, I found the same results; that is, the more I increased the number of terms K, the more the interest rate curves fluctuated (figure 6). In this project I illustrated the effect of including more terms in Schaefer method on the term structure of treasury securities. I varied the number of terms K from 15 to 45. I found that the discount rate curve corresponding to $K = 15$ resembled the one corresponding to $K = 25$. However, when I increased the value of K to 30 and to 35, the oscillations worsened. A wildly fluctuating curve resulted when I increased the value of K to 45.

Fig. 5 The effect of including more conditioned terms in cubic spline method



These oscillations occur due to the following reason. I estimate the discount rates, $d(t)$, by construction of the vector $\mathbf{x} = [x_0, x_1, \ldots, x_K]'$. Each $d(t)$ is equal to the linear combination of $x_k, k = 0, 1, \ldots, K$. When we increase the value of K, we increase the number of terms in the linear combination for each $d(t)$. Consider what would happen if we continuously increase the value of K until it equals M, the total number of bonds. This optimization problem will try to estimate a set of M variables such that they minimize the sum of the squared errors from M sets of data. From a statistical point of view, it is meaningless to do so. This reason can be used to explain the fluctuations in the cubic spline approximation as well.

Fig. 6 The effect of including more than 15 terms in Schaefer method



3.4 Effect of Monotonicity Constraints

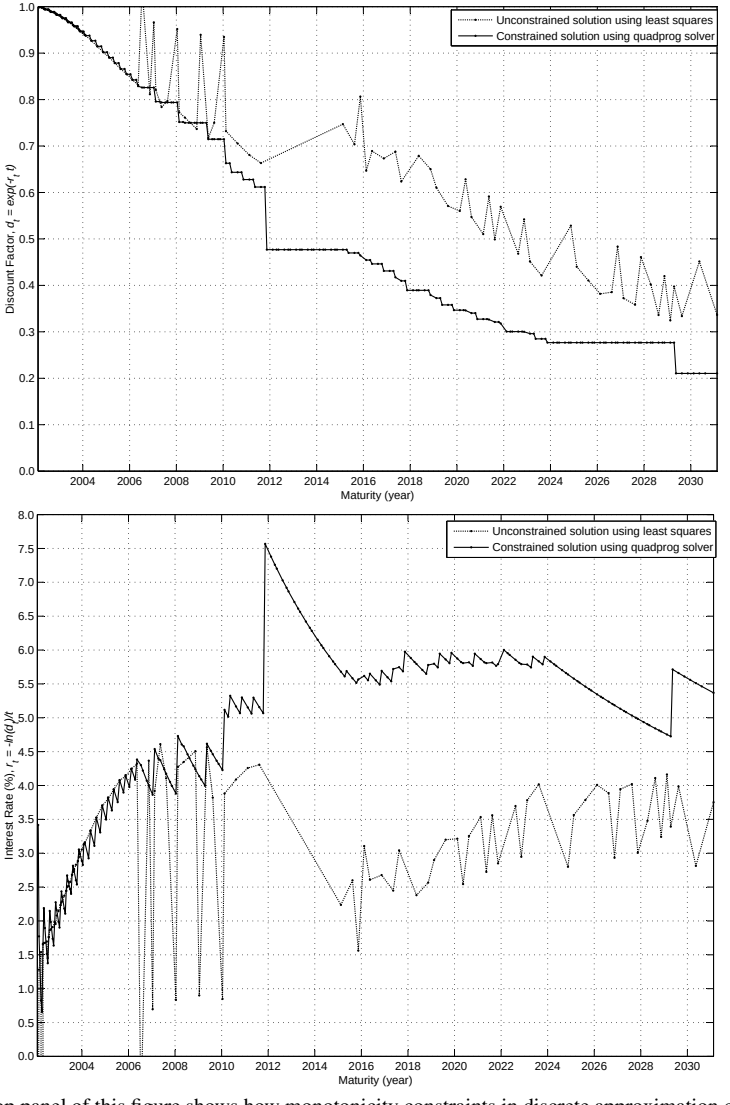
In theory, it should be the case that $d_1 \geq d_2 \geq \cdots \geq d_N$, i.e. the discount rates should be declining. Otherwise, a negative interest rate exists between two payment dates. Schaefer method guarantees that the discount rates will be declining. However, for the discrete approximation, the monotonicity constraints may or may not be enforced. I examined the discrete approximation estimates of the term structure with and without monotonicity constraints on $d(t)$. Figure 7 shows that both estimates produce discount factors that trend downward with increasing maturity, but, as expected, only when the constraints are enforced is $d(t)$ monotonically decreasing.

Figure 7 also compares the constrained and unconstrained discrete approximations to the yield curves. For early maturities - those before 2006 - the unconstrained approximation is better; it produces a smooth yield curve. In contrast, the monotonicity constraints force a series of zigzags to appear in the early part of the yield curve: when $d(t_i) = d(t_{i+1})$, then $r(t_i) > r(t_{i+1})$, which produces the downward slopes of each peak, and when $d(t_i) > d(t_{i+1})$, then $r(t_i) < r(t_{i+1})$, resulting in the small upward jumps. However, for later maturities, when yields are more difficult to predict, the unconstrained discrete approximation produces a rather noisy yield curve. The constrained solution, on the other hand, produces a yield curve with smaller fluctuations for later maturities.

Similar results were observed in Schaefer method. Figure 8 shows that the term structure of treasury securities obtained from the constrained method is smoother than the curve obtained from the unconstrained method. The unconstrained discount rates do not decrease monotonically at longer maturities which destabilizes long term interest rates.

Figure 9 demonstrates that adding monotonicity constraints to Schaefer method is important for estimation of term structures of corporate bonds and government

Fig. 7 The effect of adding monotonicity constraints in discrete approximation

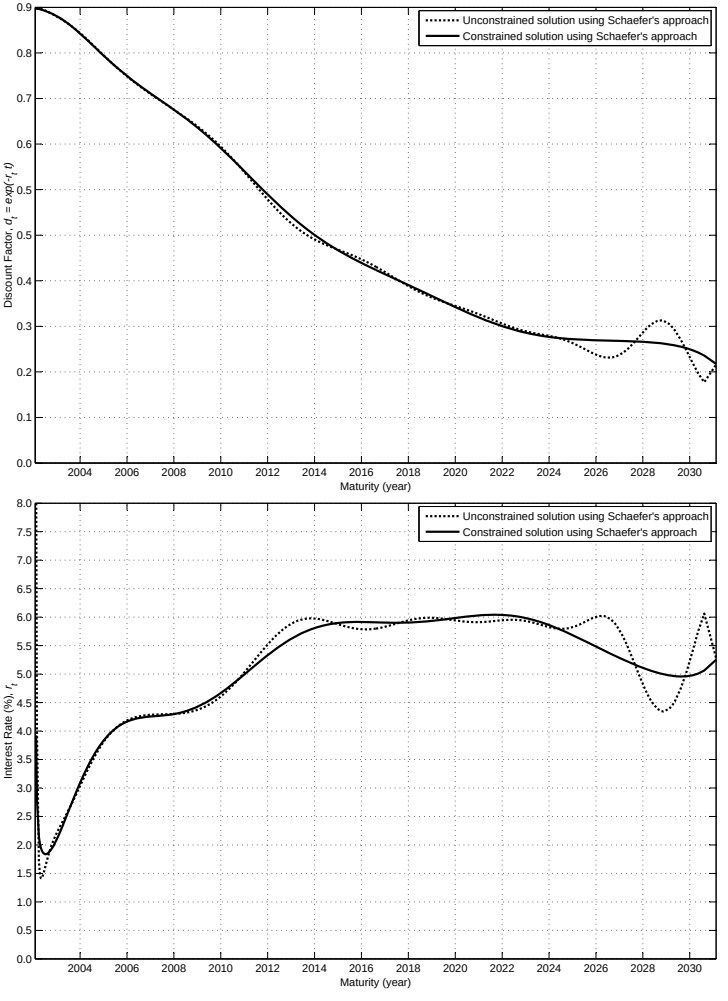


The top panel of this figure shows how monotonicity constraints in discrete approximation create L-shaped patterns in discount curve when there is no data available. This explains large deviations in term structure in the bottom panel which also shows that constrained yield curve is smoother than unconstrained one in periods when data is available.

agency bonds. Without monotonicity constraints, the term structure of these bonds is useful only for short maturities.

I also compared constrained and unconstrained versions of the cubic approximations. I decided to impose constraints on the cubic approximation in order to remove the upward slope that resulted at the end of the discount factor curve $d(t)$ when the

Fig. 8 The effect of adding monotonicity constraints in Schaefer method

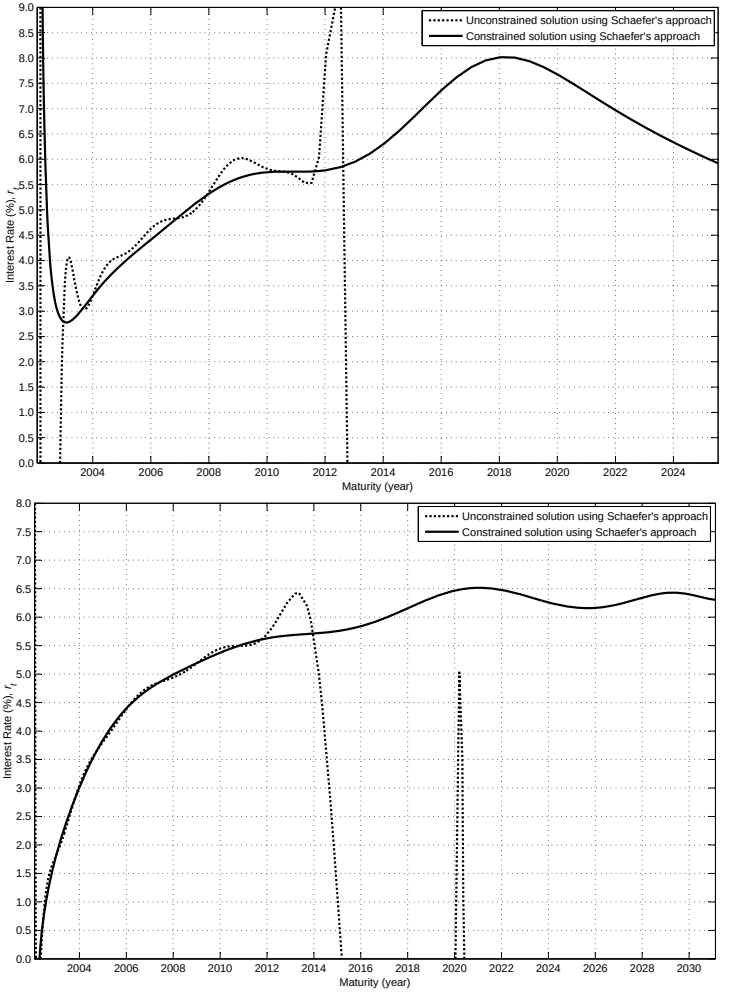


Adding monotonicity constraints in Schaefer method further reduces volatility of discount rates and interest rates. In contrast with discrete method, adding constraints to the optimization process in Schaefer method did not require much more computational effort because of less number of variables (26 in case of cubic splines compared to 211 in discrete method).

constraints were ignored. However, the constrained cubic approximation was difficult to implement. The effect of adding monotonicity constraints on discount factors was a rapid decline of $d(t)$ in the short term (figure 10). Enforcing monotonicity in the cubic approximation produces an unrealistic, nearly vertical yield curve and is thus unusable.

In terms of optimization effort, it is obvious that adding some constraints to the model will require more effort to solve for a set of solutions. In order to minimize the norm $\|\mathbf{Ad} - \mathbf{p}\|^2$, I simply use the linear least squares method. To apply this method,

Fig. 9 Constrained Schaefer method for corporate and government agency bonds



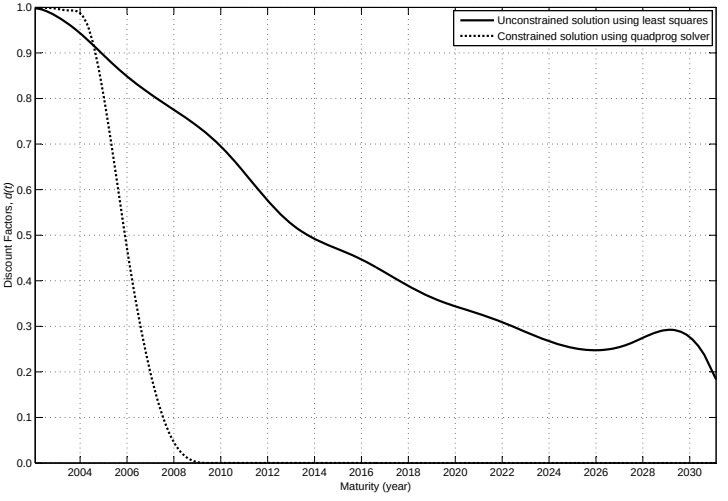
given a dependent variable, y, and a set of independent variables, $x_1, x_2, \ldots, x_N$, I try to find a linear relation by determining the set of parameters, $b_0, b_1, \ldots, b_N$, such that the sum of the squared errors is minimized. To do this, I construct a vector of dependent variables y_i, say $\mathbf{y}$, and a matrix of independent variables x_i, say $\mathbf{X}$, from the historical data and set the relations as follows:

$$\mathbf{y} = \mathbf{X}\mathbf{b} + \boldsymbol{\varepsilon}, \tag{20}$$

where $\mathbf{b}$ is the vector of parameters b_i, and $\boldsymbol{\varepsilon}$ is the vector of errors. Using the least squares method to minimize $\boldsymbol{\varepsilon}'\boldsymbol{\varepsilon}$ (the sum of the squared errors), we may calculate the solution from

$$\mathbf{b}^* = \left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{y}. \tag{21}$$

Fig. 10 The effect of adding monotonicity constraints in cubic spline method



This figure shows that we should not constrain discount factors in cubic spline method because discount factors would drop rapidly to zero.

In discrete model, I defined a vector of prices $\mathbf{p}$ and a matrix of coupon payments $\mathbf{A}$. The vector of discount rates that minimize the norm $\|\mathbf{Ad}-\mathbf{p}\|^2$ is $\mathbf{d}^* = (\mathbf{A}'\mathbf{A})^{-1}\mathbf{A}'\mathbf{p}$. For continuous models, I simply replace matrix $\mathbf{A}$ with $\mathbf{AB}$, and we have $\mathbf{x}^* = (\mathbf{B}'\mathbf{A}'\mathbf{AB})^{-1}\mathbf{B}'\mathbf{A}'\mathbf{p}$.

Although calculating the inverses of the matrices may take a long time, many software packages such as MATLAB use the interior-reflective Newton method in which each iteration involves the approximate solution of a large linear system using the method of preconditioned conjugate gradients (PCG). In practice, we solve this kind of problem in a few seconds.

When we add a set of constraints to the models, we cannot use the same formula to find the solutions. In MATLAB, when we add the inequality constraints, the problem can be solved using quadratic programming. This program uses an active set method which finds an initial feasible solution by first solving a linear programming problem. At each major iteration, a positive definite quasi-Newton approximation of the Hessian of the Lagrangian function is calculated using the BFGS method. In the case of discrete approximation, where we have a large number of variables, the time required to find monotonically constrained solutions reached the order of several minutes. However, in the case of continuous approximations, the computation time was negligible due to the small number of monotonically constrained variables.

3.5 Conclusions

In this project I experimented with various methods for approximating the term structure from different types of bonds. My results showed that the constrained Schaefer method produces smoother and more stable curves than does unconstrained Schafer's

approximation, cubic splines, or discrete approximation. Moreover, the effect of adding monotonicity constraints in Schaefer method does not significantly increase in the optimization effort as it does in the discrete approximation. My experiments showed the unreasonable results of adding monotonicity constraints on cubic spline method. Finally, I experimented with three well know parsimonious functional approximations of the term structure, the Nelson-Siegel, Svensson and Bliss exponential functions.

A Computational Appendix

In order to estimate yield curves, the computer program needs to accomplish a number of technical tasks such as scanning text files to import bonds' data, computing the cash flow payment matrix **A**, vector **t** of unique and monotonically increasing payment dates, vector **p** of cash prices. This computational appendix describes and documents my implementation of these functions in MATLAB.

A.1 Importing Bond Data From Text File

Function `tstrscan.m` is responsible for scanning text files for necessary bond data. Assuming that the input file is in the proper format, the function can import raw data on settlement date, quoted prices, coupon rates, maturity dates, coupon frequencies, and even day-count conventions for different types of bonds. Then it can compute the cash flows matrix using the specified day-count convention methodology for different types of bonds.

The following is an example of readable input text file in the proper format:

```
  ---------- begin of treasury.txt -------------
 Settle
 02/15/2002
 Coupon         Maturity     Price   Period   Basis
  5.625      12-31-2002P   103.108        2       0
  7.500      Feb-15-2005   111.131       12       1
  6.125        8-15-2007   108.239        6       2
 10.625   August-15-2015   150.485        4       3
  7.625     15/Feb-2025C   126.104        1       0
    ...              ...       ...      ...     ...
  ------------- end of treasury.txt -------------
```

To call this function we use the following syntax:

```
[Settle, Maturity, QuotedPrice, CouponRate, Period, Basis] = tstrscan('treasury.txt')
```

Note that the format of dates in my program can vary. You may want to abbreviate names of months as in 'Feb-15-2005', use a full date format with spaces such as 'February 15, 2002', or even use the letters 'C' or 'P' to distinguish between callable and putable bonds as in '15-Feb-2002C' or '15-Feb-2002P'. The only requirement is that if you use spaces then you need to use another delimiter instead of spaces. For example, you could use asterisks or commas.

Another important feature of my program is that it automatically skips columns that it does not recognize as necessary for importing. You may abbreviate column names to the first three letters. The program does not distinguish between lowercase or uppercase letters. The following column names and their abbreviations are selected for importing: **set**tlement date, **cou**pon rate, **mat**urity date, quoted **pri**ce, coupon frequency **per**iod, day-count **bas**is.

If the file contains extraneous columns aside from those indicated in the table, then they are ignored. If a special delimiter is used, then it needs to be specified as an additional parameter in call of the function `tstrscan.m`. For example, let's assume that we have the following file:

```
   -------------------------- begin of test.txt --------------------------
Settlement Date
February 15, 2002
 Type of Issue   *   Size   *      Cou   *   Maturity Date       *   Price
 T-NOTE          *   200    *    5.625   *   December 31, 2002   *   103.108
 T-BOND          *   200    *    7.500   *   Feb 15, 2005        *   111.131
 T-NOTE (5YR)    *   200    *    6.125   *   August 8 2007P      *   108.239
 T-NOTE          *   200    *   10.625   *   Aug 15 2015         *   150.485
 ...             *   ...    *      ...   *   ...                 *   ...
--------------------------- end of test.txt ---------------------------
```

It contains two extra columns, column names contain spaces, and delimiter is asterisk. To import this file we need to call the function `tstrscan.m` by specifying the delimiter '*':

```
[Settle,Maturity,QuotedPrice,CouponRate,Period,Basis] = tstrscan('test.txt','*')
```

The first two columns will not be imported because the program does not consider them necessary for subsequent analysis.

A.2 Constructing Cash Flows Matrix

One very important tool for estimating term structures is a program that does all the necessary cash flow and time mapping given bond parameters. I resorted to the `cfamounts` function from MATLAB Financial Toolbox for its functionality and ability to work with different types of bonds and day-count conventions. Once we know the exact time dates and corresponding cash flow amounts, we can proceed to construct the payment matrix **A**.

In previous section, I described how to use `tstrscan.m` function to import bond data from a text file. Now we use the result of this import as parameters for the `tstrprep.m` function:

```
[Settle,Maturity,QuotedPrice,CouponRate,Period,Basis] = tstrscan(FileName);
[p,A,s,t] = tstrprep(Settle,Maturity,QuotedPrice,CouponRate,Period,Basis);
```

The `tstrprep.m` function returns a vector of cash prices (**p**), a matrix of cash flows (**A**), a vector of settlement dates in serial date number format (**s**), and a vector of cash flow dates in serial date number format (**t**).

Input parameters are as follows:

Settle	Settlement date (must be earlier than or equal to Maturity).
Maturity	A vector of bonds' maturity dates in serial date number format.
QuotedPrice	A vector of quoted (clean) prices.
CouponRate	(Optional) A vector of percentage numbers indicating the annual percentage rate used to determine the coupons payable on a bond. Default is vector of zeros.
Period	(Optional) Coupons per year of the bond. A vector of integers (0, 1, 2, 3, 4, 6, and 12). Default is vector of 2's (two coupons per year).
Basis	(Optional) Day-count basis of the bond. A vector of integers: 0 = actual/actual, 1=30/360, 2=actual/360, 3=actual/365. Default is vector of zeros (actual/actual).

Note that CouponRate must be specified in percentages, not in decimals.

Continuing the example from the previous section, let's construct cash flows payment matrix **A** for a large set of 4462 municipal bonds using function `tstrprep.m`:

```
>> [P,A,S,T] = tstrprep(Settle,Maturity,QuotedPrice,CouponRate,Period,Basis);
>> whos
  Name               Size     Bytes   Class
  Settle              1x1         8   double array
  Maturity         4462x1     35696   double array
  QuotedPrice      4462x1     35696   double array
  CouponRate       4462x1     35696   double array
  Period           4462x1     35696   double array
  Basis            4462x1     35696   double array
  p                4462x1     35696   double array
  A             4462x1104   1457152   sparse array
  s                   1x1         8   double array
  t                1104x1      8832   double array
```

As you can see, large matrix **A** contains cash flows for 4462 bonds and 1104 time dates. It occupies 1.4 Mb of memory space while the full A matrix would occupy 37.6 Mb:

```
>> A_full = full(A); whos
  Name          Size      Bytes   Class
  A         4462x1104    1457152   sparse array
  A_full    4462x1104   39408384   double array
```

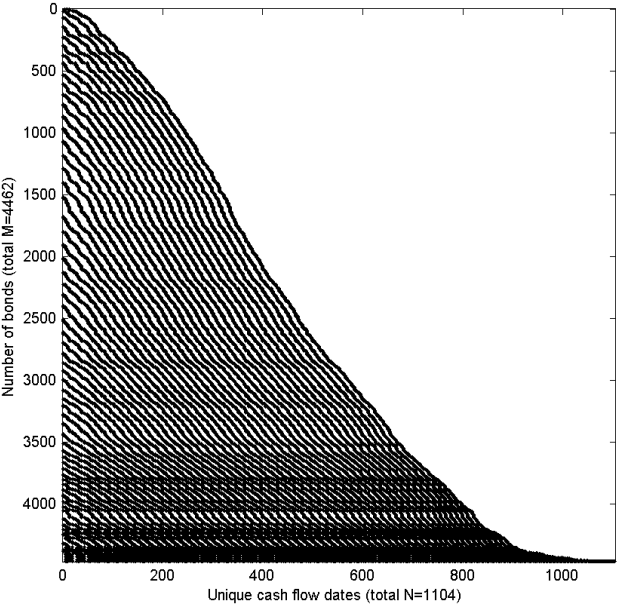
To visualize the sparsity pattern of matrix **A**, we can use the `spy(A)` command (figure 11).

The cash flows matrix conforms to the expected sparsity pattern: as the total number of bonds increases, the total number of unique cash flow dates also increases. Note the increasing effect of adding longer-term maturity bonds on total number of unique dates.

A.3 Quadratic Programming Issues

In order to add monotonicity constraints and solve quadratic programming problem, we have to convert the parameters in each model to match the quadratic programming function in MATLAB. The quadratic

Fig. 11 Sparcity Pattern of Cash Flows Matrix A



programming function `quadprog`in MATLAB requires parameters in the following form:

$$\begin{array}{l} \min\limits_{\mathbf{x}} \frac{1}{2}\mathbf{x}'\mathbf{H}\mathbf{x} + \mathbf{f}'\mathbf{x} \\ \text{s.t. } \mathbf{M}\mathbf{x} \leq \mathbf{z} \end{array} \tag{22}$$

where $\mathbf{H}$ and $\mathbf{M}$ — matrices, $\mathbf{f}, \mathbf{z}$ and $\mathbf{x}$ — vectors.

Therefore we need to put matrices and vectors of the models in the above format; namely, we have to define matrices $\mathbf{H}$ and $\mathbf{M}$ and vectors $\mathbf{f}$ and $\mathbf{z}$.

Let's put objective function $\|\mathbf{A}\mathbf{d} - \mathbf{p}\|^2$ of discrete model in the following form:

$$\begin{aligned} \|\mathbf{A}\mathbf{d} - \mathbf{p}\|^2 &= (\mathbf{A}\mathbf{d} - \mathbf{p})'(\mathbf{A}\mathbf{d} - \mathbf{p}) = (\mathbf{d}'\mathbf{A}' - \mathbf{p}')(\mathbf{A}\mathbf{d} - \mathbf{p}) \\ &= \mathbf{d}'\mathbf{A}'\mathbf{A}\mathbf{d} - \mathbf{d}'\mathbf{A}'\mathbf{p} - \mathbf{p}'\mathbf{A}\mathbf{d} + \mathbf{p}'\mathbf{p} \\ &= \mathbf{d}'(\mathbf{A}'\mathbf{A})\mathbf{d} - 2\mathbf{p}'\mathbf{A}\mathbf{d} + \mathbf{p}'\mathbf{p} \\ &= \frac{1}{2}\mathbf{d}'(2\mathbf{A}'\mathbf{A})\mathbf{d} + (-2\mathbf{A}'\mathbf{p})'\mathbf{d} + \mathbf{p}'\mathbf{p} \end{aligned}$$

Therefore, $\mathbf{H} = 2\mathbf{A}'\mathbf{A}$ and $\mathbf{f} = -2\mathbf{A}'\mathbf{p}$.

For the continuous approximation, we want to minimize $\|\mathbf{A}\mathbf{d} - \mathbf{p}\|^2 = \|\mathbf{A}\mathbf{B}\mathbf{x} - \mathbf{p}\|^2$. Hence we simply replace the matrix $\mathbf{A}$ with the product of matrices $\mathbf{A}$ and $\mathbf{B}$ and have the following results: $\mathbf{H} = 2(\mathbf{A}\mathbf{B})'\mathbf{A}\mathbf{B}$ and $\mathbf{f} = -2(\mathbf{A}\mathbf{B})'\mathbf{p}$. Then we set the variables vector $\mathbf{x}$ equal to vector $\mathbf{d}$, which completes the transformation.

To add monotonicity constraints on decreasing discount factors $\mathbf{d} = [d_1, d_2, d_3, \ldots, d_N]'$ in discrete approximation, i.e. $d_1 \geq d_2 \geq d_3 \geq \cdots \geq d_N$, we construct such $(N-1) \times N$ matrix $\mathbf{M}$ and $(N-1) \times 1$ vector $\mathbf{z}$ that $\mathbf{M}\mathbf{d} \leq \mathbf{z}$:

$$\mathbf{M} = \begin{pmatrix} -1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 1 & 0 & \cdots & 0 \\ 0 & 0 & -1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix} \text{ and } \mathbf{z} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Similarly, we can add monotonicity constraints on $d(t)$ in the cubic splines model. Since we estimate $\mathbf{d} = \mathbf{B}\mathbf{x}$, the matrix $\mathbf{M}$ is equal to its product with matrix $\mathbf{B}$ from the model. Vector $\mathbf{z}$ is a vector of zeros.

In Schaefer method we need the nonnegative $d(t)$ constraints, while their monotonous decay is guaranteed by model specification. Therefore, the only constraint I impose is $d(1) = \mathbf{b}(1)\mathbf{x} \geq 0$, where $\mathbf{b}(1)$ — last N-th row of matrix $\mathbf{B}$ (see formula 11). Therefore, in Schaefer method matrix $\mathbf{M} = -\mathbf{b}(1)$, and vector $\mathbf{z}$ is zero.

References

Bliss, R. (1997). Testing Term Structure Estimation Methods. In P. Boyle, G. Pennacchi, and P. Ritchken (Ed.), *Advances in Futures and Options Research*, 197–231.

Litzenberger, R.H., & Rolfo, J. (1984). An International Study of Tax Effects on Government Bonds. *The Journal of Finance*, 39(1), 1-22.

Nelson, C. R., & Siegel, A. F. (1987). Parsimonious Modeling of Yield Curves. *The Journal of Business*, 60(4), 473.

Schaefer, S. M. (1981). Measuring a Tax-Specific Term Structure of Interest Rates in the Market for British Government Securities. *The Economic Journal*, 91(362), 415.

Söderlind, P., & Svensson, L. (1997). New techniques to extract market expectations from financial instruments. *Journal of Monetary Economics*, 40(2), 383-429.