Occasional Paper No.2

# On Numerical Calculation Programs of American-type Options Using GAUSS Codes

By

Kenji MIYAZAKI
Hosei University

Makoto SAITO
Osaka University

September 1998

The Japan Statistics Research Institute

Hosei University, Tokyo

# On numerical calculation programs of American-type options using GAUSS codes

Kenji MIYAZAKI
Hosei University

Makoto SAITO
Osaka University

September 17, 1998

**Abstract**

This note presents the package of numerical calculation programs for the valuation of American-type options using the GAUSS code which is one of the most popular program languages among economists and econometricians. The package includes the valuation of American-type put options on securities, the term structure model, and American-type option-embedded bonds. While a part of the program package is included in hard copies in the final section, the entire package is freely available in the web site.

# 1 Introduction

Numerical calculation methods play a vital role in the derivation of pricing of options and interest-derivatives, in particular American-type and exotic options. Meanwhile the idea of American-type options has recently been applied to investment theory (see Dixit and Pindyck [2]), and become an essential concept for both finance theory and macroeconomics. This note presents GAUSS codes for the numerical calculation of American-type options and explains how to use these codes.

While these codes presented here are the by-products from our completed and progressing projects including Miyazaki and Saito [6], they may serve not merely as a technical appendix for our paper, but also for the following purposes. First, our package includes the calculation codes for standard option pricing and term structure problems such that they may be used as concrete numerical examples for advanced undergraduate and graduate courses. Second, with minor modifications, these codes are applicable to a wide range of option pricing and investment decision problems. Third, our programs are written in the GAUSS code which is one of the most popular matrix program languages among economists and econometricians. In this sense, our program package is complementary to that using the C language (Duffie [3]) or the Basic language (Kijima et al. [5]).

This note is organized as follows. Section 2 demonstrates the numerical algorithm for the valuation of American-type options using the finite-difference method. Our examples include American-type put options on securities (Karatzas [4]), the term structure model (Cox, Ingersoll and Ross [1] and Vacisek [7]), and American-type option-embedded bonds (Miyazaki and Saito [6]). Section 3 provides the GAUSS codes for the numerical derivation of American-type options.

# 2 Finite-Difference Method and American-type Options

## 2.1 General Formula

Suppose a financial derivative whose price depends on both time $t$ and the value of its underlying security $x(t)$. $x(t)$ is characterized by the following Ito process:

$$dx = \mu(x,t)dt + \sigma(x,t)dB_t,$$

where $\mu(x,t)$ is a drift part, $\sigma(x,t)$ is a diffusion part, and $B_t$ is a standard Brownian motion. The above derivative security promises to pay dividends $D(x,t)$ at time $t$ and a terminal value $g(x)$ at time $T$.

When spot rates are determined by $r(x,t)$, the time $t$ value of the security $F(x,t)$ is obtained by solving the following Cauchy problem: given real-valued functions $r, g, \mu, \sigma$, and $D$ on $\mathbf{R} \times [0,T]$, find a function $F : \mathbf{R} \times [0,T) \to \mathbf{R}$ solving the partial differential equation:

$$\frac{\partial F(x,t)}{\partial t} + \frac{1}{2}\sigma(x,t)^2\frac{\partial^2 F(x,t)}{\partial x^2} + \mu(x,t)\frac{\partial F(x,t)}{\partial x} + D(x,t) = r(x,t)F(x,t), \qquad (1)$$

with the terminal condition:

$$F(x,T) = g(x). \tag{2}$$

See Duffie [3] for the detailed derivation and regularity conditions that guarantee the existence and uniqueness of solutions.

### 2.1.1 Finite-difference method

Given the difficulty with obtaining analytical solutions in the field of finance theory, numerical calculation methods frequently play a vital role in finding solutions. Such methods include the finite-difference method, the lattice method, and the Monte Carlo simulation method. Our paper [6] adopts the finite-difference method among these, mainly because this method is fairly convenient for the valuation of American-type options. This subsection briefly explains the finite-difference method.

The main idea of this method for solving equation (1) given the terminal condition (2) is to choose grids as follows

$$\{(x_i, t_j) : i \in \{1, \ldots, N+1\}, j \in \{1, \ldots, M+1\}\},$$

and to find an approximate solution of (1) given (2) in the form of an $N+1 \times M+1$ matrix $F$ whose $(i,j)$-element $F_{i,j}$ is to be the approximation of $F(x_i, t_j)$ where $t_1 = 0$ and $t_{M+1} = T$. $x_i - x_{i-1} = \Delta x$ and $t_i - t_{i-1} = \Delta t$ are constant for all $i, j$. $\Delta x$ and $\Delta t$ are called the mesh sizes of grids.

In the literature, either the explicit method, the implicit method, or the Crank-Nicholson method is employed for the approximation of the above function (1). In what follows, we describe the Crank-Nicholson method which is the most accurate and stablest among them[1]. In this method, the first and second derivatives of $F(x,t)$ are approximated as follows:

$$\begin{aligned}
\left.\frac{\partial F(x,t)}{\partial t}\right|_{x=x_i, t=t_j} &\simeq \frac{F_{i,j+1} - F_{i,j}}{\Delta t}, \\
\left.\frac{\partial F(x,t)}{\partial x}\right|_{x=x_i, t=t_j} &\simeq \frac{1}{2}\left(\frac{F_{i+1,j} - F_{i-1,j}}{2\Delta x} + \frac{F_{i+1,j+1} - F_{i-1,j+1}}{2\Delta x}\right), \\
\left.\frac{\partial^2 F(x,t)}{\partial x^2}\right|_{x=x_i, t=t_j} &\simeq \frac{1}{2}\left(\frac{F_{i+1,j} - 2F_{i,j} + F_{i-1,j}}{(\Delta x)^2} + \frac{F_{i+1,j+1} - 2F_{i,j+1} + F_{i-1,j+1}}{(\Delta x)^2}\right).
\end{aligned}$$

Substituting these approximations into equation (1) leads to the following expression: for $1 < i < N+1$,

$$a_{i,j} F_{i-1,j} + b_{i,j} F_{i,j} + c_{i,j} F_{i+1,j} = h_{i,j} \tag{3}$$

at $(x_i, t_j)$ where

$$a_{i,j} = \frac{\Delta t}{4\Delta x}\left(-\mu(x_i, t_j) + \frac{\sigma(x_i, t_j)^2}{4(\Delta x)}\right),$$

[1] For the other methods, see Kijima et al. [5].

$$
\begin{aligned}
b_{i,j} &= -1 - \frac{\sigma(x_i,t_j)^2\Delta t}{2(\Delta x)^2} - r(x_i,t_j)\Delta t, \\
c_{i,j} &= \frac{\Delta t}{4\Delta x}\left(\mu(x_i,t_j) + \frac{\sigma(x_i,t_j)^2}{4(\Delta x)}\right), \\
h_{i,j} &= -a_{i,j}F_{i-1,j+1} + d_{i,j}F_{i,j+1} - c_{i,j}F_{i+1,j+1} + e_{i,j}, \\
d_{i,j} &= -1 + \frac{\sigma(x_i,t_j)^2\Delta t}{2(\Delta x)^2}, \\
e_{i,j} &= -D(x_i,t_j)\Delta t.
\end{aligned} \tag{4}
$$

Since equation (3) cannot be defined at either $i = 1$ or $N+1$, the following expression may be chosen alternatively:

$$
b_{1,j}F_{1,j} - c_{1,j}F_{2j} = h_{1,j}, \quad a_{N+1,j}F_{N,j} + b_{N+1,j}F_{N+1,j} = h_{N+1,j}. \tag{5}
$$

Using concrete examples, we will later explain how to choose the above coefficients $b_{1,j}$, $c_{1,j}$, $h_{1,j}$, $b_{N+1,j}$, $c_{N+1,j}$, and $h_{N+1,j}$.

Given equations (3) and (5), a backward difference equation is obtained for the columns $F_1, F_2, \ldots, F_{M+1}$ of $F$ by

$$
A_jF_j = h_j, \tag{6}
$$

with the terminal condition:

$$
F_{i,M+1} = g(x_i), \tag{7}
$$

where $A_j$ is the tridiagonal matrix given by

$$
A_j = \begin{pmatrix}
b_{1j} & c_{1j} & 0 & 0 & 0 & \cdots & 0 \\
a_{2j} & b_{2j} & c_{2j} & 0 & 0 & \cdots & 0 \\
0 & a_{3j} & b_{3j} & c_{3j} & 0 & \cdots & 0 \\
\vdots & & & & \ddots & & \vdots \\
0 & \cdots & 0 & 0 & a_{N,j} & b_{N,j} & c_{N,j} \\
0 & \cdots & 0 & 0 & 0 & a_{N+1,j} & b_{N+1,j}
\end{pmatrix},
$$

and where $h_j$ is the vector with $i$-th element $h_{i,j}$. Then, the approximated function $F_{i,j}$ is obtained by determining the coefficients in $A = j$ for all $j$.

The above Crank-Nicholson method is summarized as the following steps:

**Step 1:** Fix $F_{M+1}$ according to the terminal condition (7).

**Step 2**: Set $j = M$.

**Step 3**: Do the following procedure (**Step 4**, **5**, and **6**) until $j = 1$.

**Step 4**: Compute $h_j$ from (4) and (5).

**Step 5**: Compute $F_j$ by solving (6).

**Step 6**: Reset $j = j - 1$, and go back to **Step 3**.

In the above steps, the solution of (6) can be obtained by utilizing the LU decomposition, which will be described later.

### 2.1.2 American-type options

This subsection explores additional considerations required by the valuation of American-type options. The owner of American-type options has the right to receive a payoff $\Omega(x,t)$ at time $t$. S/he exercises this right whenever the value of not-exercising is dominated by that of exercising. The value function of an American-type option $F$ before exercising ($x > x^*(t)$ or $x < x^*(t)$) satisfies the partial differential equation (1) with the terminal condition (2), the value matching condition:

$$F(x^*(t),t) = \Omega(x^*(t),t),$$

and the smooth pasting condition:

$$\left.\frac{\partial F(x,t)}{\partial x}\right|_{x=x^*(t)} = \left.\frac{\partial \Omega(x,t)}{\partial x}\right|_{x=x^*(t)}$$

In the above case, the numerical algorithm follows:

**Step 1:** Fix $F_{M+1}$ according to the terminal condition (7).

**Step 2:** Set $j = M$.

**Step 3:** Do the following procedure (**Step 4**, **5**, **6** and **7**) until $j = 1$.

**Step 4:** Compute $h_j$ from (4) and (5).

**Step 5:** Compute $\hat{F}_j$ by solving $A_j\hat{F}_j = h_j$.

**Step 6:** Compute $F_j$ by $F_{i,j} = \max[\hat{F}_{i,j}, \Omega(x_i,t_j)]$.

**Step 7:** Reset $j = j - 1$, and go back to **Step 3**.

Because the main property of the finite-difference method is to transform states from continuous variables to discrete ones by making grids, the smooth pasting condition, which is the consideration of continuity at a critical point, is not considered explicitly in the above algorithm. Nevertheless, **Step 6** in the above algorithm guarantees the smooth pasting condition in a close approximation.

### 2.1.3 LU decomposition

Finally, the LU decomposition is explained as below. Consider an unknown $n$-vector $x$ of a set of liner equations $Ax = y$ where $y$ is a known $n$-vector and $A$ is a known and non-singular $n \times n$ matrix. The LU decomposition is a simple, but powerful algorithm for solving $x$ without directly inverting matrix $A^{-1}$.

The most important property of matrix $A$ is that it has the following tridiagonal structure:

$$A = \begin{bmatrix} a_1 & c_1 & 0 & \cdots & 0 \\ b_1 & a_2 & c_2 & & \vdots \\ 0 & b_2 & a_3 & \ddots & 0 \\ \vdots & & \ddots & \ddots & c_{n-1} \\ 0 & \cdots & 0 & b_{n-1} & a_n \end{bmatrix}.$$

With consideration of the above tridiagonal matrix[2], the numerical algorithm is constructed as follows.

**Step 1:** Set $\{u_i\}_{i=1}^n$ as

$$u_1 = a_1; \quad u_i = a_i - \frac{c_{i-1}b_{i-1}}{u_{i-1}}; \quad i = 2, 3, \ldots, n.$$

**Step 2:** Obtain $\{z_i\}_{i=1}^n$ by starting $z_1 = y_1$ and calculating

$$z_i = y_i - \frac{b_{i-1}}{u_{i-1}} z_{i-1},$$

in a forward step of $i (i = 2, \ldots, n)$.

**Step 3:** Obtain $\{x_i\}_{i=1}^n$ by starting $x_n = z_n/u_n$ and calculating

$$x_i = (z_i - c_{i+1}x_{i+1})/u_i$$

in a backward step of $i (i = n-1, \ldots, 1)$.

## 2.2 American-Type Put Options on Securities

As one of classical examples of American-type options, we present the numerical procedure for the American-type put options on securities. Suppose that the owner of this options has the right to sell a security at a constant exercise price $K$ before the option matures at time $T$. A spot rate $r$ is constant over time, and security prices $S$ follow the geometric Brownian motion. Under the risk-neutral (martingale) measure, the security price follows

$$\frac{dS}{S} = rdt + \sigma dB_t,$$

where $\sigma$ is constant and $B_t$ is a standard Brownian motion.

The value function $f$ of this option is found by solving the following partial differential equation at the continuation region $S > S^*(t)$ (see Karatzas [4].):

$$\frac{\partial f(S,t)}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f(S,t)}{\partial S^2} + rS\frac{\partial f(S,t)}{\partial S} = rf(S,t) \tag{8}$$

with the terminal conditions:

$$f(S,T) = \max[K - S, 0],$$

$$\lim_{S\to\infty} f(S,t) = 0, \tag{9}$$

$$\lim_{S\to 0} f(S,t) = Ke^{-r(T-t)}, \tag{10}$$

[2]Such tridiagonal matrixes frequently appear in the numerical calculation of finance theory.

the value matching condition:

$$f(S^*(t), t) = \max[K - S^*(t), 0],$$

and the smooth pasting condition:

$$\left.\frac{\partial f(S,t)}{\partial S}\right|_{S=S^*(t)} = -1.$$

$S$ is transformed as follows, thereby making the range of $S$ a finite interval:

$$x(S) \equiv \frac{1}{1+\alpha S}, \; S \in [0, \infty), \tag{11}$$

for some $\alpha > 0$. In this paper, $\alpha$ is chosen such that the initial stock price $S_0$ is centered or $1/(1+\alpha S_0) = 1/2$. Inverting (11) leads to

$$S(x) = \frac{1-x}{\alpha x}, \; x \in (0, 1].$$

Letting $F(t, x) \equiv f(t, S(x))$, we replace the above conditional partial differential equation with the following:

$$\frac{\partial F(x,t)}{\partial t} + \frac{1}{2}P(x)\frac{\partial^2 F(x,t)}{\partial x^2} + Q(x)\frac{\partial F(x,t)}{\partial x} = rF(x,t), \tag{12}$$

with

$$\begin{aligned}
&F(x,T) = \max[K - (1-x)/\alpha x, 0], \\
&\lim_{x\to 0} F(x,t) = 0, \\
&\lim_{x\to 1} F(x,t) = Ke^{-r(T-t)}, \\
&F(x^*(t),t) = \max[K - (1-x^*(t))/\alpha x^*(t), 0], \\
&\left.\frac{\partial F(x,t)}{\partial x}\right|_{x=x^*(t)} = -1,
\end{aligned}$$

where

$$\begin{aligned}
P(x) &= \sigma^2 x^2 (1-x)^2, \\
Q(x) &= \sigma^2 x (1-x)^2 - rx(1-x).
\end{aligned}$$

Under this transformation, the range of $x$ is (0,1], and the following grids are constructed:

$$\{(x_i, t_j) : i \in \{1, \ldots, N+1\}, j \in \{1, \ldots, M+1\}\},$$

where

$$\begin{aligned}
x_i &= (i-1)\Delta x, \quad i = 1, \ldots, N+1; \quad \Delta x = \frac{1}{N}, \\
t_j &= (j-1)\Delta t, \quad j = 1, \ldots, M+1; \quad \Delta t = \frac{T}{M},
\end{aligned}$$

$x_1 = 0$, $x_N = 1$, $t_1 = 0$, and $t_M = T$. $F_{i,j}$ is denoted as $F(x_i, t_j)$. Substituting the approximate derivatives of $F$ in the previous subsection into (12), we can obtain at $(x_i, t_j)$, for $i = 2, \ldots, N, j = 1, \ldots, M$,

$$a_i F_{i-1,j} + b_i F_{i,j} + c_i F_{i+1,j} = -a_i F_{i-1,j+1} + d_i F_{i,j+1} - c_i F_{i+1,j+1},$$

where

$$\begin{aligned} a_i &= \frac{\Delta t}{4\Delta x}\left(-Q(x_i) + \frac{P(x_i)}{\Delta x}\right), \\ b_i &= -1 - P(x_i)\frac{\Delta t}{2(\Delta x)^2} - r\Delta t, \\ c_i &= \frac{\Delta t}{4\Delta x}\left(Q(x_i) + \frac{P(x_i)}{\Delta x}\right), \\ d_i &= -1 + P(x_i)\frac{\Delta t}{2(\Delta x)^2}. \end{aligned}$$

$F_{1,j} = 0$ and $F_{N+1,j} = K\exp(-r(T - t_j))$ are given by the terminal conditions.

Now, we define an $M \times N$-matrix, $H = \{h_{i,j}\}_{j=1,\ldots,M}^{i=1,\ldots,N}$ as

$$\begin{aligned} h_{i,j} &= -a_{i+1}F_{i,j+1} + d_{i+1}F_{i+1,j+1} - c_{i+1}F_{i+2,j+1},\ i = 1, \ldots, N-1, \\ h_{N,j} &= K\exp(-r(T - t_j)), \end{aligned}$$

and an $N$-tridiagonal matrix, $A$ as

$$A \equiv \begin{bmatrix} b_2 & c_2 & 0 & \cdots & 0 \\ a_3 & b_3 & c_3 & & \vdots \\ 0 & a_4 & b_4 & \ddots & 0 \\ \vdots & & \ddots & \ddots & c_N \\ 0 & \cdots & \cdots & 0 & 1 \end{bmatrix}.$$

Given the above notations, the structure of $G \equiv \{G_{i,j}\}_{j=1,\ldots,M}^{i=1,\ldots,N} \equiv \{F_{i+1,j}\}_{j=1,\ldots,M}^{i=1,\ldots,N}$ is simplified as

$$AG = H.$$

The $j$-th column of $G, H$ is denoted as $G_j, H_j$.

The numerical calculation of $F$ is summarized as follows:

**Step 1**: Start with the terminal condition:

$$F_{i,M+1} = \max[K - (1 - x_i)/\alpha x_i, 0], \quad i = 1, \ldots, N+1.$$

**Step 2**: Set $j = M$.

**Step 3**: Do the following procedure (**Step 4**, **5**, **6**, **7**, and **8**) until $j = 1$.

**Step 4**: Compute $H_j$ by the above definition.

**Step 5**: Compute $\hat{G}_j$ by solving $A\hat{G}_j = H_j$.

**Step 6**: Compute $G_j$ by

$$G_{i,j} = \max[\hat{G}_{i,j}, \max[K - (1 - x_{i+1})/\alpha x_{i+1}, 0]].$$

**Step 7**: Compute $F_j$ by

$$F_{1,j} = 0, \; F_{i,j} = G_{i-1,j}, \quad i = 2, \ldots, N+1$$

**Step 8**: Reset $j = j - 1$, and go back to **Step 3**.

## 2.3 Term Structure Model

### 2.3.1 CIR process

This subsection presents the numerical calculation for the term structure or the pricing of standard discount bonds. Consider a standard discount bond which pays out one unit of principal at time $T$. A spot rate is assumed to evolve according to the Cox-Ingersoll-Ross process ([1], hereafter the CIR process):

$$dr = \eta(\bar{r} - r)dt + \sigma\sqrt{r}dB_t,$$

where $\bar{r}$ and $\sigma$ are constant. Spot rates take the range of $[0, \infty)$ under the CIR process. An alternative spot rate process (the Ornstein-Uhlenbeck process) will be considered in the next subsection.

Given the risk neutrality assumption, the time $t$ value of this discount bond $f(r,t)$ is obtained by solving the following differential equation:

$$\frac{\partial f(r,t)}{\partial t} + \frac{1}{2}r\sigma^2\frac{\partial^2 f(r,t)}{\partial r^2} + \eta(\bar{r} - r)\frac{\partial f(r,t)}{\partial r} = rf(r,t), \tag{13}$$

with the terminal conditions:

$$f(r,T) = 1, \tag{14}$$

$$\left.\frac{\partial f(r,t)}{\partial t}\right|_{r=0} + \eta\bar{r}\left.\frac{\partial f(r,t)}{\partial r}\right|_{r=0} = 0, \tag{15}$$

$$\lim_{r\to\infty} f(r,t) = 0. \tag{16}$$

Equation (15) is derived from (13) with $r = 0$. (16) implies that the bond with sufficient high spot rate is valueless.

The yield to maturity on the above bond $\tilde{r}_T$ is represented by

$$\tilde{r}_T = -\frac{1}{T}\log(f(r_0, 0)), \tag{17}$$

where $r_0$ is the initial spot rate.

As in the case of American-type put options on securities, the following transformation is made for the process of $r(t)$ for making state variables move in a finite interval:

$$x(r) \equiv \frac{1}{1+\alpha r}, \; r(x) = \frac{1-x}{\alpha x},$$

for some $\alpha > 0$. Let $F(t,x) \equiv f(t, r(x))$. Hence, the above partial differential equation (13) is replaced with

$$\frac{\partial F(x,t)}{\partial t} + \frac{1}{2}P(x)\frac{\partial^2 F(x,t)}{\partial x^2} + Q(x)\frac{\partial F(x,t)}{\partial x} = R(x)F(x,t), \tag{18}$$

and the above conditions with

$$\begin{aligned} & F(x,T) = 1, \\ & \left.\frac{\partial F(x,t)}{\partial t}\right|_{x=1} + Q(1)\left.\frac{\partial F(x,t)}{\partial x}\right|_{x=1} = 0, \\ & \lim_{x\to 0} F(x,t) = 0, \end{aligned}$$

where

$$\begin{aligned} P(x) &= \sigma^2 \alpha x^3 (1-x), & (19) \\ Q(x) &= \left\{\eta\left(\frac{1-x}{\alpha x} - \bar{r}\right) + \frac{1}{2}\sigma^2(1-x)\right\}\alpha x^2, & (20) \\ R(x) &= \frac{1-x}{\alpha x}F. \end{aligned}$$

Choosing the grid setting and the approximate derivatives in the same manner as in the previous subsection lead to, at $(x_i, t_j)$, for $i = 2, \ldots, N, j = 1, \ldots, M$,

$$a_i F_{i-1,j} + b_i F_{i,j} + c_i F_{i+1,j} = -a_i F_{i-1,j+1} + d_i F_{i,j+1} - c_i F_{i+1,j+1},$$

where

$$\begin{aligned} a_i &= \frac{\Delta t}{4\Delta x}\left(-Q(x_i) + \frac{P(x_i)}{\Delta x}\right), \\ b_i &= -1 - P(x_i)\frac{\Delta t}{2(\Delta x)^2} - R(x_i)\Delta t, \\ c_i &= \frac{\Delta t}{4\Delta x}\left(Q(x_i) + \frac{P(x_i)}{\Delta x}\right), \\ d_i &= -1 + P(x_i)\frac{\Delta t}{2(\Delta x)^2}. \end{aligned}$$

With $i = 1$ $(x = 0)$, the spot rate $r(t)$ is infinite. Hence, one of the terminal conditions yields

$$F_{1,j} = 0, \;\; j = 1, \ldots, M.$$

At $i = N+1$ ($x = 1$) the following approximation is made:

$$\left.\frac{\partial F(x,t)}{\partial t}\right|_{x=1,t=t_j} \simeq \frac{F_{N+1,j+1} - F_{N+1,j}}{\Delta t},$$
$$\left.\frac{\partial F(x,t)}{\partial x}\right|_{x=1,t=t_j} \simeq \frac{F_{N+1,j} - F_{N,j}}{\Delta x}. \tag{21}$$

Given the above approximation, for $j = 1, \ldots, M$,

$$a_{N+1}F_{N,j} + b_{N+1}F_{N+1,j} = -F_{N+1,j},$$

where

$$a_{N+1} = -Q(1)\frac{\Delta t}{\Delta x},\ b_{N+1} = -1 + Q(1)\frac{\Delta t}{\Delta x}.$$

An $M \times N$-matrix, $H = \{h_{i,j}\}_{j=1,\ldots,M}^{i=1,\ldots,N}$ is defined as

$$\begin{aligned} h_{i,j} &= -a_{i+1}F_{i,j+1} + d_{i+1}F_{i+1,j+1} - c_{i+1}F_{i+2,j+1},\ i = 1, \ldots, N-1, \\ h_{N,j} &= -F_{N+1,j+1}, \end{aligned}$$

and an $N$-tridiagonal matrix, $A$ as

$$A \equiv \begin{bmatrix} b_2 & c_2 & 0 & \cdots & 0 \\ a_3 & b_3 & c_3 & & \vdots \\ 0 & a_4 & b_4 & \ddots & 0 \\ \vdots & & \ddots & \ddots & c_N \\ 0 & \cdots & 0 & a_{N+1} & b_{N+1} \end{bmatrix}.$$

Given these notations, the structure of $G \equiv \{G_{i,j}\}_{j=1,\ldots,M}^{i=1,\ldots,N} \equiv \{F_{i+1,j}\}_{j=1,\ldots,M}^{i=1,\ldots,N}$ is simplified as

$$AG = H.$$

The $j$-th column of $G, H$ is defined as $G_j, H_j$.

The calculation procedure of $F$ is as follows:

**Step 1**: Start with the terminal condition:

$$F_{i,M+1} = 1, \quad i = 1, \ldots, N+1.$$

**Step 2**: Set $j = M$.

**Step 3**: Do the following procedure (**Step 4**, **5**, **6**, and **7**) until $j = 1$.

**Step 4**: Compute $H_j$ by the above definition.

**Step 5**: Compute $G_j$ by solving $AG_j = H_j$.

**Step 6**: Compute $F_j$ by

$$F_{1,j} = 0,\ F_{i,j} = G_{i-1,j}, \quad i = 2, \ldots, N+1$$

**Step 7**: Reset $j = j - 1$, and go back to **Step 3**.

## 2.4 American-Type Option-Embedded Bonds

This subsection presents the numerical procedure for the valuation of put and call options embedded in standard discount bonds. We here take for example the model of Miyazaki and Saito [6] that analyzes the Japanese public financial system. On the liability side of this system, the postal savings account offers put-option-embedded bonds to depositors, while the asset side has callable loans (call-option-embedded bonds) which allow borrowers to repay principals prior to maturity. While the setup of this model is discussed in great detail in Miyazaki and Saito [6], this note mainly explains the numerical procedure for solving this model. Since the procedure adopted here has common features with those discussed in the previous subsections (2.2 and 2.3), only differences among these models are emphasized in this subsection.

### 2.4.1 Put-option-embedded bonds (postal savings account)

The postal savings account works as a put-option-embedded bond as shown below. On the one hand, this account offers a ten-year fixed rate deposit. The fixed rate $\tilde{r}$ is determined by $\min[\tilde{r}_{10} - 0.005, 0.95\tilde{r}_3]$, where $\tilde{r}_T(T = 3, 10)$ is the yield on the $T$-year discount bond given by equation (17). On the other hand, in addition to this fixed rate feature, this account allows depositors to cancel the account prior to ten-year maturity with mild penalty.

Thanks to the above 'put like' option, depositors are able to switch the initial contract to a higher-yield bond contract when spot rates become high enough. Considering the actual feature of the postal savings account carefully, Miyazaki and Saito [6] assume that depositors can switch to standard discount bonds prior to maturity, and that depositors have to pay mild penalty before three years pass since the initial contract was made.

Given the above setup, at the continuation regions $r < r^*(t)$, the time $t$ value of the postal savings, denoted by $f(r, t)$, should satisfy the partial differential equation (13) with the terminal conditions (14)-(16), the value matching condition:

$$f(r^*(t), t) = \Omega(t)$$

where

$$\begin{aligned} \Omega(t) &= \exp\{-\tilde{r}(10-t)\} \exp\left\{-\frac{2}{45}\tilde{r}(t-3)^2\right\}, \quad 0 \le t < 3, \\ &= \exp\{-\tilde{r}(10-t)\}, \quad 3 \le t \le 10, \end{aligned}$$

and the smooth pasting condition:

$$\left.\frac{\partial f(r,t)}{\partial r}\right|_{r=r^*(t)} = 0.$$

The specification of $\Omega(t)$ considers the above-mentioned mild penalty for early (less than three years) cancellation.

The corresponding algorithm for the valuation of the postal savings account requires minor modifications of the case of the CIR standard discount bond (Section 2.3) as follows:

**Step 1**: Start with the terminal condition:

$$F_{i,M+1} = 1, \quad i = 1, \ldots, N+1.$$

**Step 2**: Set $j = M$.

**Step 3**: Do the following procedure (**Step 4**, **5**, **6**, **7**, and **8**) until $j = 1$.

**Step 4**: Compute $H_j$ by the above definition.

**Step 5**: Compute $\hat{G}_j$ by solving $A\hat{G}_j = H_j$.

**Step 6**: Compute $G_j$ by $G_{i,j} = \max[\hat{G}_{i,j}, \Omega(t_j)]$.

**Step 7**: Compute $F_j$ by

$$F_{1,j} = 0, \; F_{i,j} = G_{i-1,j}, \quad i = 2, \ldots, N+1$$

**Step 8**: Reset $j = j - 1$, and go back to **Step 3**.

### 2.4.2 Call-option-embedded bonds (callable loans)

The callable loan contract considered here is almost a symmetric case of the postal savings account. In Miyazaki and Saito [6], the borrowing rate on this loan contract is fixed at the level of the yield on the ten-year discount bond. Borrowers are allowed to call loans without any penalty prior to maturity (a ten-year maturity is assumed),

With minor modifications of the model of the postal savings account, it is possible to numerically value this callable bond. That is, **Step 6** in the above algorithm should be replaced with

**Step 6'**: Compute $G_j$ by $G_{i,j} = \min[\hat{G}_{i,j}, \exp(-\tilde{r}_{10}(10 - t_j))]$ where $\tilde{r}_{10}$ is the ten-year bond yield.

## 2.5 OU process

Finally, we consider the case of the bond valuation for an alternative spot rate process. That is, spot rates evolve according to the Ornstein-Uhlenbeck process (hereafter the OU process):

$$dr = \eta(\bar{r} - r)dt + \sigma dB_t.$$

One problem with the OU process is that spot rates may be negative[3]. However, negative spot rates can be ruled out by replacing equation (15) with

$$\left.\frac{\partial f(r,t)}{\partial r}\right|_{r=0} = 0,$$

[3]Using the OU process, Vasicek derives the term structure of interest rates in a closed form allowing for negative spot rates.

in the case of OU process. This condition may be regarded as the 'smooth pasting like' condition at $r$=0.

Considering the above condition, the numerical procedure has to make two modifications to that with the CIR process. First, the expression (19) and (20) are replaced with

$$\begin{aligned} P(x) &= \sigma^2\alpha^2x^4 \\ Q(x) &= \left\{\eta\left(\frac{1-x}{\alpha x}-\bar{r}\right)\alpha x^2+\frac{1}{2}\sigma^2\alpha^2x^3\right\}. \end{aligned}$$

Second, the terminal condition at $r=0$ is formulated as follows. Noting $x=1/(1+\alpha r)$ and $F(x,t)=f(r,t)$, the terminal condition at $r=0$ implies

$$\left.\frac{\partial F(x,t)}{\partial x}\right|_{x=1}=0.$$

Hence, for $j=1,\ldots,M$,

$$F_{N,j}-F_{N+1,j}=0$$

is derived from (21). Consequently, the matrix $H$ and $A$ in the CIR case are replaced with

$$\begin{aligned} h_{i,j} &= -a_{i+1}F_{i,j+1}+d_{i+1}F_{i+1,j+1}-c_{i+1}F_{i+2,j+1},\ i=1,\ldots,N-1, \\ h_{N,j} &= 0, \end{aligned}$$

and

$$A=\begin{bmatrix} b_2 & c_2 & 0 & \cdots & 0 \\ a_3 & b_3 & c_3 & & \vdots \\ 0 & a_4 & b_4 & \ddots & 0 \\ \vdots & & \ddots & \ddots & c_N \\ 0 & \cdots & 0 & 1 & -1 \end{bmatrix}.$$

Utilizing the algorithm of the CIR case with the above modifications, we can obtain the numerical solution $F$ with the OU case.

# 3 Code Examples

This section presents the GAUSS code package for the asset pricing models described above: American-type put options on securities, American-type option-embedded bonds with the CIR process, and American-type option-embedded bonds with the OU process.

This package consists of the following codes:

`main_bs.prg`, `main_cir.prg`, `main_ou.prg`,
`bs_put.prc`, `cir_m.prc`, `cir_all.prc`, `ou_m.prc`, `ou_all.prc`,
`lu.prc`.

Code "*.prg" is a main program of each example, and Code "*.prc" is a subroutine procedure called by main programs. lu.prc is the procedure of the LU decomposition which is utilized in all examples.

Among the main programs, main_bs.prg is the program that calculates the value of American-type put options on securities. main_cir.prg is the main program that calculates the value of discount bonds with options (put/call) when spot rates follow the CIR, while main_ou.prg regards the case for the OU process. The latter two programs also include the term structure models or the valuation of standard discount bonds.

The dependency among the above codes is summarized as follows:

```
main_bs.prg   ← lu.prc + bs_put.prc,
main_cir.prg  ← lu.prc + cir_m.prc   + cir_all.prc,
main_ou.prg   ← lu.prc + ou_m.prc    + ou_all.prc.
```

This section includes only main_bs.prg and main_cir.prg in hard copies in order to save the manuscript space. Other related programs as well as the codes listed here are available in the following web site:

http://prof.mt.tama.hosei.ac.jp/~miya_ken/research.htm .

One minor change required for running the above codes is that d:\gauss\gwork\options\[4] must be replaced with a chosen working directory[5].

Anyone may use or modify freely the above codes with appropriate acknowledgement and citation. While the authors have paid every possible attention in writing these codes, they do not have any responsibility for potential problems caused by the codes.

[4] These terms are found in #include phrase of main_bs.prg and main_cir.prg.

[5] After running these codes, the error message "Math coprocessor exceptions: Division by zero" always appears. This message, however, has nothing to do with any numerical results.

## 3.1 lu.prc

```
/*****************
*LU decomposition*
*****************/
proc lu_decomp(a,b,c,nn);
    local i,u;
    u=zeros(nn,1);
    u[1]=a[1];
    i=2;
    do until i>nn;
        u[i]=a[i]-b[i-1]*c[i-1]/u[i-1];
        i=i+1;
    endo;
    retp(u);
endp;

proc lu_calc(a,b,c,u,h,nn);
    local i,y,g;
    y=zeros(nn,1);
    g=zeros(nn,1);
    y[1]=h[1];
    i=2;
    do until i>nn;
        y[i]=h[i]-b[i-1]*y[i-1]/u[i-1];
        i=i+1;
    endo;
    g[nn]=y[nn]/u[nn];
    i=nn-1;
    do until i<1;
        g[i]=(y[i]-c[i]*g[i+1])/u[i];
        i=i-1;
    endo;
    retp(g);
endp;
```

## 3.2 bs_put.prc

```
/*****************
*** bs_put.prc ***
******************/
/*grid setting*/
a=zeros(n+1,1);
b=zeros(n+1,1);
c=zeros(n,1);
d=zeros(n,1);
i=2;
do until i>n;
    p=sigma^2*(1-x[i])^2*x[i]^2;
    q=sigma^2*x[i]*(1-x[i])-r*x[i]*(1-x[i]);
    a[i]=delta_t*(-q+p/delta_x)/(4*delta_x);
    b[i]=-1-p*delta_t/(2*delta_x^2)-r*delta_t;
    c[i]=delta_t*(q+p/delta_x)/(4*delta_x);
    d[i]=-1+p*delta_t/(2*delta_x^2);
    i=i+1;
endo;
b[n+1]=1;
aa=zeros(n,1);
bb=zeros(n-1,1);
cc=zeros(n-1,1);
i=1;
do until i>n-1;
    aa[i]=b[i+1];
    bb[i]=a[i+2];
    cc[i]=c[i+1];
    i=i+1;
endo;
aa[n]=b[n+1];
u=lu_decomp(aa,bb,cc,n);
f =zeros(n+1,m+1);
f1=zeros(n+1,m+1);
S_star=zeros(m,1);

/*initial condition*/
i=2;
do until i>n+1;
    temp=K-(1-x[i])/(alpha*x[i]);
    if temp<0;
        temp=0;
    endif;
        f[i,m+1] =temp;
    f1[i,m+1]=temp;
```

```
    i=i+1;
endo;

/*main loop*/
j=m;
do until j<1;
    h =zeros(n,1);
    h1=zeros(n,1);
    i=1;
    do until i>n-1;
        h[i]=-a[i+1]*f[i,j+1]+d[i+1]*f[i+1,j+1]-c[i+1]*f[i+2,j+1];
        h1[i]=-a[i+1]*f1[i,j+1]+d[i+1]*f1[i+1,j+1]-c[i+1]*f1[i+2,j+1];
        i=i+1;
    endo;
    h[n]=K*exp(-r*(TT-t[j]));
    h1[n]=K*exp(-r*(TT-t[j]));
    g =lu_calc(aa,bb,cc,u,h,n);
    g1=lu_calc(aa,bb,cc,u,h1,n);
    f[2:n+1,j]=g;
    i=n+1;
    do until i<2;
        if g1[i-1] .< v[i];
            S_star[j]=SS[i];
            g1[i-1]=v[i];
        endif;
        i=i-1;
    endo;
    f1[2:n+1,j]=g1;
    j=j-1;
endo;
```

## 3.3 main_bs.prg

```
/**************
* main_bs.prg *
**************/
/* variable setting */
new;
S0=100;
K=100;
alpha = (2-1)/S0;
r=0.05;
sigma=0.25;
n=300;
delta_x = 1/n;
x = seqa(0,delta_x,n+1);
/*SS = (1-x)./(alpha*x);*/
i=2;
do until i>n+1;
    SS[i] = (1-x[i])/(alpha*x[i]);
    i=i+1;
    endo;
SS[1]=2*SS[2];
tt= 1;
uy = 300;
/* unit:  year */
m = uy*tt;
delta_t = tt/m;
t = seqa(0,delta_t,m+1);
v=zeros(n+1,1);
i=2;
do until i>n+1;
    temp=K-(1-x[i])/(alpha*x[i]);
    if temp<0;
        temp=0;
    endif;
    v[i] =temp;
    i=i+1;
endo;
/*including LU decomposition procedure*/
#include d:\gauss\gwork\options\lu.prc;
/*including sub numerical caluculation procedure*/
#include d:\gauss\gwork\options\bs_put.prc;
S_star = S_star|K;
/**********
* Output **
**********/
```

```
print "=============================================";
print "S(0) = ";;S0;
print "K = ";;K;
print "r = ";;r;
print "sigma = ";;sigma;
print "n = ";;n;
print "m = ";;m;
print "=============================================";
print "Value of European put option = ";;f[floor(n/2)+1,1];
print "Value of American put option = ";;f1[floor(n/2)+1,1];
print "=============================================";
/*****************
*Output(graphics)**
*****************/
/* Number of Passing Years for graph.prg */
/*passy=0;*/
passy=0;
ay=passy;
ay=uy*ay+1;
fg = f1[.,ay]~f[.,ay];
library pgraph;
graphset;
_pdate="";
_plwidth=10;
xlabel("stock price");
ylabel("option value");
title("American put option value
(" $+ ftos((10-passy),"%*.*lf",2,0)
$+"years remaining, initial price at "
$+ ftos(S0,"%*.*lf",1,2)$+")");
_plegstr = "American option\000European option";
_plegctl = 2~5~5~4;
xy(SS[floor(n/4):n+1],fg[floor(n/4):n+1,.]);
graphset;
xlabel("passing years");
ylabel("critical stock price");
title("Critical Values of American Options
(initial price at " $+ ftos(S0,"%*.*lf",1,2)$+")");
xy(t,S_star);
end;
stop;
```

## 3.4 cir_m.prc

```
/***************
*** cir_m.prc ***
***************/
/*grid setting*/
a=zeros(n+1,1);
b=zeros(n+1,1);
c=zeros(n,1);
d=zeros(n,1);
i=2;
do until i>n;
    p=sigma^2*alpha*x[i]^3*(1-x[i]);
    q=alpha*x[i]^2*(eta*((1-x[i])/(alpha*x[i])-r_bar)+sigma^2*(1-x[i])/2);
    r=(1-x[i])/(alpha*x[i]);
    a[i]=delta_t*(-q+p/delta_x)/(4*delta_x);
    b[i]=-1-p*delta_t/(2*delta_x^2)-r*delta_t;
    c[i]=delta_t*(q+p/delta_x)/(4*delta_x);
    d[i]=-1+p*delta_t/(2*delta_x^2);
    i=i+1;
endo;
q=-eta*alpha*r_bar;
a[n+1]=-q*delta_t/delta_x;
b[n+1]=-1+q*delta_t/delta_x;
aa=zeros(n,1);
bb=zeros(n-1,1);
cc=zeros(n-1,1);
i=1;
do until i>n-1;
    aa[i]=b[i+1];
    bb[i]=a[i+2];
    cc[i]=c[i+1];
    i=i+1;
endo;
aa[n]=b[n+1];
u=lu_decomp(aa,bb,cc,n);
f=zeros(n+1,m+1);
/*initial condition*/
f[.,m+1]=ones(n+1,1);
/*main loop*/
j=m;
do until j<1;
    h=zeros(n,1);
    i=1;
    do until i>n-1;
    h[i]=-a[i+1]*f[i,j+1]+d[i+1]*f[i+1,j+1]-c[i+1]*f[i+2,j+1];
```

```
        i=i+1;
    endo;
    h[n]=-f[n+1,j+1];
    g=lu_calc(aa,bb,cc,u,h,n);
    f[2:n+1,j]=g;
    j=j-1;
endo;
r1=-ln(f[floor(n/2)+1,1])/tt;
```

## 3.5 cir_all.prc

```
/****************
** cir_all.prc **
****************/
/*grid setting*/
a=zeros(n+1,1);
b=zeros(n+1,1);
c=zeros(n,1);
d=zeros(n,1);
i=2;
do until i>n;
    p=sigma^2*alpha*x[i]^3*(1-x[i]);
    q=alpha*x[i]^2*(eta*((1-x[i])/(alpha*x[i])-r_bar)+sigma^2*(1-x[i])/2);
    r=(1-x[i])/(alpha*x[i]);
    a[i]=delta_t*(-q+p/delta_x)/(4*delta_x);
    b[i]=-1-p*delta_t/(2*delta_x^2)-r*delta_t;
    c[i]=delta_t*(q+p/delta_x)/(4*delta_x);
    d[i]=-1+p*delta_t/(2*delta_x^2);
    i=i+1;
endo;
q=-eta*alpha*r_bar;
a[n+1]=-q*delta_t/delta_x;
b[n+1]=-1+q*delta_t/delta_x;
aa=zeros(n,1);
bb=zeros(n-1,1);
cc=zeros(n-1,1);
i=1;
do until i>n-1;
    aa[i]=b[i+1];
    bb[i]=a[i+2];
    cc[i]=c[i+1];
    i=i+1;
endo;
aa[n]=b[n+1];
u=lu_decomp(aa,bb,cc,n);
f =zeros(n+1,m+1);
f1=zeros(n+1,m+1);
f2=zeros(n+1,m+1);
ff1=zeros(n+1,m+1);
ff2=zeros(n+1,m+1);
r_star1=zeros(m,1);
r_star2=zeros(m,1);
/*initial condition*/
f[.,m+1] =ones(n+1,1);
f1[.,m+1]=ones(n+1,1);
```

```
f2[.,m+1]=ones(n+1,1);
ff1[.,m+1]=ones(n+1,1);
ff2[.,m+1]=ones(n+1,1);
/*main loop*/
j=m;
do until j<1;
    h =zeros(n,1);
    h1=zeros(n,1);
    h2=zeros(n,1);
    i=1;
    do until i>n-1;
        h[i]=-a[i+1]*f[i,j+1]+d[i+1]*f[i+1,j+1]-c[i+1]*f[i+2,j+1];
        h1[i]=-a[i+1]*f1[i,j+1]+d[i+1]*f1[i+1,j+1]-c[i+1]*f1[i+2,j+1];
        h2[i]=-a[i+1]*f2[i,j+1]+d[i+1]*f2[i+1,j+1]-c[i+1]*f2[i+2,j+1];
        i=i+1;
    endo;
    h[n]=-f[n+1,j+1];h1[n]=-f1[n+1,j+1];h2[n]=-f2[n+1,j+1];
    g =lu_calc(aa,bb,cc,u,h,n);
    g1=lu_calc(aa,bb,cc,u,h1,n);
    g2=lu_calc(aa,bb,cc,u,h2,n);
    f[2:n+1,j]=g;
    ff1[2:n+1,j]=g1;
    ff2[2:n+1,j]=g2;
    i=2;
    do until i>n+1;
        if g1[i-1] .<= v1[j];
            r_star1[j]=rr[i];
            g1[i-1]=v1[j];
        endif;
        i=i+1;
    endo;
    i=n+1;
    do until i<2;
        if g2[i-1] .>= v2[j];
            r_star2[j]=rr[i];
            g2[i-1]=v2[j];
        endif;
        i=i-1;
    endo;
    f1[2:n+1,j]=g1;
    f2[2:n+1,j]=g2;
    j=j-1;
endo;
```

## 3.6 main_cir.prg

```
/***************
* main_cir.prg *
***************/
/* variable setting */
new;
r0=0.05;
uy = 52;
/* unit:  year */
/* 52 grids = 1 year */
eta = 0.0514;
sigma = 0.0492;
r_bar = 0.0499;
alpha = (2-1)/r0;
n=100;
delta_x = 1/n;
x = seqa(0,delta_x,n+1);
/*rr = (1-x)./(alpha*x);*/
i=2;
do until i>n+1;
    rr[i] = (1-x[i])/(alpha*x[i]);
    i=i+1;
    endo;
rr[1]=2*rr[2];
/*****************************************
including LU decomposition procedure
returns procedures:
lu_decomp, lu_calc
*****************************************/
#include d:\gauss\gwork\options\lu.prc;
/* derivation of interest of 3 year discount bond */
tt= 3;
m = uy*tt;
delta_t = tt/m;
t = seqa(0,delta_t,m+1);
/***********************************************
including sub numerical caluculation procedure
returns a value:  r1
r1 == annual yield of 3 year discount bond
***********************************************/
#include d:\gauss\gwork\options\cir_m.prc;
r_03=r1;
r_a=r1*0.95;
/* derivation of interest of 10 year discount bond */
tt= 10;
```

```
m = uy*tt;
delta_t = tt/m;
t = seqa(0,delta_t,m+1);
/***********************************************
including sub numerical caluculation procedure
returns a value:  r1
r1 == annual yield of 10 year discount bond
***********************************************/
#include d:\gauss\gwork\options\cir_m.prc;
r_10=r1;
r_b = r1-0.005;
if r1<0;
    r1=0;
    print "Warning !!  yield too low!!";
endif;
/*determination of r_tilde*/
if r_a < r_b;
    r_tilde=r_a;
else;
    r_tilde=r_b;
endif;
/* r_tilde1(put) */
r_tilde1=r_tilde;
/* r_tilde2(call) */
r_tilde2=r_10;
v1=zeros(m+1,1);
v2=zeros(m+1,1);
i=1;
do until i>3*uy+1;
    r_tilde1_p=r_tilde1-(2/45)*r_tilde1*(t[i]-3)^2;
    v1[i]=exp(-r_tilde1*tt)*exp(r_tilde1_p*t[i]);
    r_tilde2_p=r_tilde2;
    v2[i]=exp(-r_tilde2*tt)*exp(r_tilde2_p*t[i]);
    i=i+1;
endo;
do until i>m+1;
    v1[i]=exp(-r_tilde1*(tt-t[i]));
    v2[i]=exp(-r_tilde2*(tt-t[i]));
    i=i+1;
endo;
/******************
Output(Parameter) *
******************/
print "=======================";
print "r(0) = ";;r0;
print "eta = ";;eta;
```

```
print "sigma = ";;sigma;
print "r_bar = ";;r_bar;
print "r(3 years) = ";;r_03;
print "r(10 years)= ";;r_10;
print "r_tilde = ";;r_tilde;
print "n = ";;n;
print "==================================================";
/*re-setting*/
tt=10;
m = uy*tt;
delta_t = tt/m;
t = seqa(0,delta_t,m+1);
/***********
main program
************/
/*****************************************************
including main numerical caluculation procedure
returns values:
f
r_star1, f1, ff1 (put)
r_star2, f2, ff2 (call)
*****************************************************/
#include d:\gauss\gwork\options\cir_all.prc;
r_star1 = r_star1|r_tilde1;
r_star2 = r_star2|r_tilde2;
/***************************
Output(Numerical Solution) *
***************************/
print "Value of bond/loan without options ";;f[floor(n/2)+1,1];
print "Value of put-embedded bond ";;f1[floor(n/2)+1,1];
print "Value of callable loan ";;f2[floor(n/2)+1,1];
print "==============================================";
/***************
Output(graphics)
****************/
/* Number of Passing Years for graph.prg */
/*passy=0;*/
passy=3;
ay=passy;
ay=uy*ay+1;
fg = f[.,ay];
f1g = f1[.,ay];
f2g = f2[.,ay];
ff = fg~f1g~f2g;
library pgraph;
graphset;
```

```
_pdate="";
_plwidth=10;
xlabel("spot rate");
ylabel("bond price");
title("Prices of Option-Embedded Bonds and Loans
(" $+ ftos((10-passy),"%*.*lf",2,0)$+
" years remaining, initial rate at " $+
ftos(r0,"%*.*lf",1,2)$+", 1.00 at maturity)");
_plegstr =
"bond/loan without options\000put-embedded bond\000call-embedded loan";
_plegctl = 2~5~1~1;
xy(rr[floor(n/4):n+1],ff[floor(n/4):n+1,.]);
graphset;
_pdate="";
_plwidth=10;
xlabel("passing years");
ylabel("critical spot rate");
title("Critical Values of American Options
(initial rate at " $+ ftos(r0,"%*.*lf",1,2)$+")");
_plegstr =
"put-embedded bond\000call-embedded loan";
_plegctl = 2~5~4~3;
xy(t,r_star1~r_star2);
end;
stop;
```

# References


[1] Cox, J. C., J. E. Ingersoll, and S. A. Ross, 1985, "A theory of the term structure of interest rates," *Econometrica* 53, 385-408.

[2] Dixit, A. and R. Pindyck, 1994, *Investment under uncertainty.* Princeton University Press.

[3] Duffie, D., 1996, *Dynamic asset pricing theory.* Princeton University Press.

[4] Karatzas, I., 1988, "On the pricing of American options," *Applied mathematics and optimization* 17, 37-60.

[5] Kijima, M., I. Izumi, and Y. Ohmi, 1996, *Introduction to financial engineering, vol. III: Numerical methods.* Nikka-Giren (in Japanese).

[6] Miyazaki, K., and M. Saito, 1998, "On the market risk involved in the public financial system in Japan: A theoretical and numerical investigation," mimeographed.

[7] Vasicek, O., 1977, "An equilibrium characterization of the term structure," *Journal of financial economics* 5, 177-188.