Convertible Bond Valuation

Zhao Xiaoyang
Risk Management Institute
National University of Singapore
June 19, 2017

A convertible bond is a corporate debt security that gives the holder the right to exchange future coupon payments and principal repayment for a prescribed number of shares of equity. Thus, it has both an equity part and a fixed-income part. Beside, it may contain some additional features, such as callability and puttability. The analytical solutions for convertible bond are not available and the accurate numerical or simulation methods are also difficult since some features require N out of M days above trigger prices. This feature leads the problem to be path-dependent and the numerical or simulation methods to be high dimensions, which will cost a long time.

In this document, we use a one-touch model to approximate the N out of M features, we replace original trigger price B_c and B_p by an equivalent trigger price B_c^* and B_p^*. Then we use finite difference method to solve the related PDE under Black-Scholes assumptions.

1 PDE Representation

The finite difference method documented here is mainly based on [1] and [2].

Under the real world measure, the underlying price S_t is assumed to follow a stochastic process

$$dS_t = \mu S_t d_t + \sigma S_t dB_t, \tag{1}$$

where μ is the drift rate, σ is the volatility of the underlying price and B_t is the standard Brownian motion.

We assume that a convertible bond has the following contractual features:

• A continuous (time-dependent) put provision with an exercise price of B_p, during the put-allowed period, if the stock price is below B_p at least N days in the last M days, the holder can sell the bond for price B_p.

• A continuous (time-dependent) conversion provision. During the convertible-allowed period, the bond can be converted to $\kappa(t)$ shares.

• A continuous (time-dependent) call provision. During the convertible-allowed period, if the stock price is above the trigger at least N days in the last M days, the issuer can call the bond for price $B_c > B_p$. However, the holder can convert the bond if it is called.

Besides, Let S^+ be the stock price immediately after default, and S^- be the stock price right before default. We will assume that

$$S^+ = S^-(1-\eta),$$

and p is the default rate, $0 < R < 1$ is the recovery factor. According to risk neutral argument, the value of the convertible bond is given by

$$MV - pmax(\kappa S(1-\eta), RB) = 0,$$

subject to the constraints

$$V \geq max(B_p, \kappa S),$$
$$V \leq max(B_c, \kappa S),$$

where $\mathcal{M}$ is operator given by:

$$\mathcal{M}V = -\frac{\partial V}{\partial t} - \left((r + p\eta - q)S\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} - (r+p)V\right). \tag{2}$$

The solution of this system can be derived by a linear complementarity problem (LCP). Define $V = V(S,t)$, the LCP can be described as follows,

- $B_c > \kappa S$,

$$\begin{cases} \mathcal{M}V - pmax(\kappa S(1-\eta), RB) = 0 \\ V - max(B_p, \kappa S) \geq 0 \\ V - B_c \leq 0, \end{cases} \begin{cases} \mathcal{M}V - pmax(\kappa S(1-\eta), RB) \geq 0 \\ V - max(B_p, \kappa S) = 0 \\ V - B_c \leq 0, \end{cases} \begin{cases} \mathcal{M}V - pmax(\kappa S(1-\eta), RB) \leq 0 \\ V - max(B_p, \kappa S) \geq 0 \\ V - B_c = 0. \end{cases}$$

- $B_c \leq \kappa S, \quad V = \kappa S.$

(3)

where B is the bond component of convertible bond. And the terminal condition is

$$V(S,T) = max(F, \kappa S).$$

Based on [2], the convertible can spilt to bond component B and the equity component C, and $V = B + C$. B and C satisfy the following LCP, separately.

$$\begin{cases} \mathcal{M}C - pmax(\kappa S(1-\eta) - RB, 0) = 0 \\ C - (\kappa S - B) \geq 0 \\ C - max(B_c, \kappa S - B) \leq 0, \end{cases} \quad or \quad \begin{cases} \mathcal{M}C - pmax(\kappa S(1-\eta) - RB, 0) \geq 0 \\ C - (\kappa S - B) = 0 \end{cases}$$

$$or \quad \begin{cases} \mathcal{M}C - pmax(\kappa S(1-\eta) - RB, 0) \leq 0 \\ C - max(B_c, \kappa S - B) = 0. \end{cases}$$

(4)

$$\begin{cases} \mathcal{M}B - RpB = 0 \\ B - (B_p - C) \geq 0 \\ B - B_c \leq 0, \end{cases} \quad or \quad \begin{cases} \mathcal{M}B - RpB \leq 0 \\ B - B_c = 0, \end{cases}$$

$$or \quad \begin{cases} \mathcal{M}B - RpB \geq 0 \\ B = B_p - C. \end{cases} \tag{5}$$

Using the penalty method, define $V_1^* = max(B_p, \kappa S)$ and $V_2^* = max(B_c, \kappa S)$, the LCP system (3) can be approximated by the following nonlinear PDE

$$\begin{cases} \mathcal{M}V - pmax(\kappa S(1-\eta), RB) + \rho max(V_1^* - V, 0) - \rho max(V - V_2^*, 0) = 0, & 0 \leq t < T \\ V(S,T) = max(F, \kappa S), & t = T. \end{cases} \tag{6}$$

where $\rho \to +\infty$.

2 PDE Discretization

To approximate the solution of the above PDEs, we use the finite difference method(FDM) for both spatial and temporal discretization. We use the change of variables $\tau = T - t$ to transform the PDEs from forward time to backward time. Divide time direction by N grids and price direction by I grids.

$$\begin{aligned} V_i^{n+1} - V_i^n =&(1-\theta)\left(\frac{\alpha_i+\beta_i}{2}(V_{i+1}^n - 2V_i^n + V_{i-1}^n) + \frac{\beta_i-\alpha_i}{2}(V_{i+1}^n - V_{i-1}^n) - (r+p)V_i^n dt\right) \\ &+\theta\left(\frac{\alpha_i+\beta_i}{2}(V_{i+1}^{n+1} - 2V_i^{n+1} + V_{i-1}^{n+1}) + \frac{\beta_i-\alpha_i}{2}(V_{i+1}^{n+1} - V_{i-1}^{n+1}) - (r+p)V_i^{n+1} dt\right) \\ &+pmax(\kappa S_i(1-\eta), RB_i^{n+1})dt, \end{aligned} \tag{7}$$

where $i = 1, ..., I-1$ and $n = 1, ..., N-1$,

$$\begin{cases} \alpha_i = (\frac{\sigma^2 S_i^2}{2(ds)^2} - \frac{(r+p\eta-q)S_i}{2(ds)})dt, \\ \beta_i = (\frac{\sigma^2 S_i^2}{2(ds)^2} + \frac{(r+p\eta-q)S_i}{2(ds)})dt. \end{cases}$$

And

$$\begin{aligned} B_i^{n+1} - B_i^n =&(1-\theta)\left(\frac{\alpha_i+\beta_i}{2}(B_{i+1}^n - 2B_i^n + B_{i-1}^n) + \frac{\beta_i-\alpha_i}{2}(B_{i+1}^n - B_{i-1}^n) - (r+p)(1-R)B_i^n dt\right) \\ &+\theta\left(\frac{\alpha_i+\beta_i}{2}(B_{i+1}^{n+1} - 2B_i^{n+1} + B_{i-1}^{n+1}) + \frac{\beta_i-\alpha_i}{2}(B_{i+1}^{n+1} - B_{i-1}^{n+1}) - (r+p)(1-R)B_i^{n+1} dt\right). \end{aligned} \tag{8}$$

where $i = 1, ..., I-1$ and $n = 1, ..., N-1$. When $\theta = 0$, we get the explicit method; When $\theta = 0.5$, we get the Crank-Nicolson method; When $\theta = 1$, we get the fully implicit method;

We use penalty method to solve the LCP, the penalty term is chosen as a large quantity related to *tol*, a sufficiently small tolerance level.

$$P_i^{n+1,B_p} = \begin{cases} \frac{1}{tol} & \text{if } V_i^{n+1} < V_1^*, \\ 0 & \text{if } V_i^{n+1} \geq V_1^*, \end{cases} \tag{9}$$

and

$$P_i^{n+1,B_c} = \begin{cases} -\frac{1}{tol} & \text{if } V_i^{n+1} > V_2^*, \\ 0 & \text{if } V_i^{n+1} \leq V_2^*. \end{cases} \tag{10}$$

From (7) and (8), we obtain a nonlinear system with matrix form

$$\big(\mathbb{I}-\theta\mathbb{M}_v\big)\mathbf{V}^{n+1} = (\mathbb{I}+(1-\theta)\mathbb{M}_v)\mathbf{V}^n + pmax(\kappa S(1-\eta) + R\mathbf{B}^{n+1}) + \mathbb{P}^{n+1,B_p}(\mathbf{V}_1^* - \mathbf{V}^{n+1}) + \mathbb{P}^{n+1,B_c}(\mathbf{V}^{n+1} - \mathbf{V}_2^*), \tag{11}$$

and

$$\big(\mathbb{I} - \theta\mathbb{M}_B\big)\mathbf{B}^{n+1} = (\mathbb{I} + (1-\theta)\mathbb{M}_B)\mathbf{B}^n, \tag{12}$$

$\mathbb{P}^{n+1,B_c}, \mathbb{P}^{n+1,B_p}$ is a diagonal matrix with $(\mathbb{P}^{n+1,B_c})_{ii} = P_i^{n+1,B_c}$ and $(\mathbb{P}^{n+1,B_p})_{ii} = P_i^{n+1,B_p}$. Combined with the boundary conditions, $\mathbb{M}_v$ is a $(I+1)\times(I+1)$ tri-diagonal matrix in the form

$$\mathbb{M}_v = \begin{pmatrix} -(r+p)dt & 0 & 0 & 0 & \dots \\ \alpha_1 & -(\alpha_1+\beta_1+(r+p)dt) & \beta_1 & 0 & \dots \\ 0 & \alpha_2 & -(\alpha_2+\beta_2+(r+p)dt) & \beta_2 & \dots \\ \vdots & \vdots & \ddots & \ddots & \ddots \\ 0 & 0 & \alpha_{I-1} & -(\alpha_{I-1}+\beta_{I-1}+(r+p)dt) & \beta_{I-1} \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

$\mathbb{M}_B$ is a $(I+1)\times(I+1)$ tri-diagonal matrix in the form

$$\mathbb{M}_B = \begin{pmatrix} -(r+p(1-R))dt & 0 & 0 & 0 & \dots \\ \alpha_1 & -(\alpha_1+\beta_1+(r+p(1-R))dt) & \beta_1 & 0 & \dots \\ \vdots & \vdots & \ddots & \ddots & \ddots \\ 0 & 0 & \alpha_{I-1} & -(\alpha_{I-1}+\beta_{I-1}+(r+p(1-R))dt) & \beta_{I-1} \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

For each backward recursive step, we first compute $\mathbf{B}$ by (12) and using Newton iteration to solve the nonlinear equation (11). If *tol* and N is set properly, the iteration number is generally less than 10 times.

3 Convertible Bond Features and Boundary Conditions

3.1 Boundary Conditions

When $S \to +\infty (i = I)$, (3) and (5) becomes

$$\frac{\partial^2 V}{\partial S^2} = 0, \qquad \frac{\partial^2 B}{\partial S^2} = 0.$$

Thus we have

$$V_I^n = \kappa S_I^n, \qquad B_I^N = B_I^{N-1} = \ldots = B_I^1 = 0.$$

When $S = 0$, (3) and (5) becomes

$$\frac{\partial V}{\partial t} = -(r+p)V, \qquad \frac{\partial B}{\partial t} = -(r+p(1-R))B,$$

and we can also use FDM to solve this equation, finally get the boundary condition.

3.2 Coupon Payments Adjustment

We assume the coupon payment is C_n, and $t_k^c \le t < t_{k+1}^c$, then the accrued interest on the pending coupon payment at time t is

$$AccI(t) = c_{k+1}\frac{t - t_k^c}{t_{k+1}^c - t_k^c}.$$

Usually quoted prices are clean prices, dirty prices include any accrued interest that has accumulated since the last coupon payment. The dirty call price B_c and the dirty put price B_p are given by

$$B_c(t) = B_c^{cl} + AccI(t),$$
$$B_p(t) = B_p^{cl} + AccI(t).$$

3.3 Period No Put, No Call and No convert

In the period when the Convertible Bond is not callable, we assign $B_c(t)$ a large value so that the call event is unlikely to happen. In the period when the Convertible Bond is not puttable, we assign $B_p(t)$ zero so that the put event never happens. In the period when the Convertible Bond is not convertible, we assign $\kappa(t)$ zero so that the convert event never happens.

3.4 N-of-M Triggers: An Equivalent Trigger Price

N-of-M triggers are seen in soft calls and also in contingent conversion provisions. Exact modeling of such contract provisions is extremely time consuming, requiring a prohibitive 2^M PDE grid levels for an exact solution. We model these provisions by calculating an equivalent barrier $B_c^*(t)$ such that the probability of $S(\cdot)$ exceeding the barrier $B_c(t)$ for the required N of the previous M days is equal to the probability of the $S(t)$, the stock at time t, exceeding $B_c^*(t)$. This reduces the N-of-M barrier to an equivalent 1-of-1 barrier, namely one-touch model. We then apply the relevant constraint at time t based on the equivalent barrier $B_c^*(t)$ and $B_p^*(t)$. $B_c^*(t)$ is calculated by the following equation

$$\mathbf{P}\left(\exists j \le T : \sum_{i=j-M+1}^{j} \mathbf{1}_{\{S(t_i)\ge B_c(t_i)\}} \ge N\right) = \mathbf{P}\left(\max_{i\le T}\{S(t_i)\} \ge B_c^*(t_i)\right). \tag{13}$$

B_p^* is calculated similarly:

$$\mathbf{P}\left(\exists j \le T : \sum_{i=j-M+1}^{j} \mathbf{1}_{\{S(t_i)\le B_p(t_i)\}} \ge N\right) = \mathbf{P}\left(\min_{i\le T}\{S(t_i)\} \le B_p^*(t_i)\right). \tag{14}$$

We use Montecarlo method to get this equivalent trigger price.

3.5 Stock Dividends

We assume the stock pay dividends once a year, at the middle of the year, namely when $t = X.5$, $X = 5, 4, 3, 2, 1, 0$, there will be a dividends $q * S(t)$. When paying dividends, the convertible price will be adjusted from $\frac{F}{\kappa(t)}$to $\frac{F}{\kappa(t)} - q * S(t)$.

4 Code Description and Some Numerical Results

We investigate two convertible bonds in this part, Geer and Guangqi, and we use the five days implied volatility as volatility input.

4.1 Code Description

4.1.1 The Main Code

The following code is the main code to calculate price.

```
function U = convertible_code(Nt,Ns,F,conversion_price,Smax,S0,risk_free_rate,rate_T,sigma,T,final_coupon_rate,q,p,R,eta,no_call_time,...
no_put_time,no_convert_time,Bc_star,Bp_star,theta,coupon_time_rate)
%-----Solving pde-Penalty mehtod-CN difference-Newton iteration-------
% clc;
% clear;
% tic;
%
% ---------parameter specification-------------
% Nt=400;%time direction
% Ns=200;%price direction
% F=100;%face value
% conversion_price=26.43;% convert price
% Smax=155;%upbound of stock price
% S0=31;%now stock price
% r=0.0345;%risk free rate
% sigma=1.52;% volatility
% T=3.5;% time to maturaty
% final_coupon_rate=0.08;%coupon rate in the final time
% q=0;% dividend
% p=0;%default probobility
% R=0.5;% recovery rate
% eta=0.5;%when default the stock price becomes (1-eta)*S
% no_call_time=[5.38 6];%no allow call time
% no_put_time=[3.88 6];%no allow put time
% no_convert_time=[5.38 6];%no allow convert time
```

```
% Bc_star=141.65;%callable price
% Bp_star=F*1;%puttable price
% theta=1;%implicitness parameter，0.5为CN method,1为fully implicit or backward euler method
% coupon_time_rate=[0.07 0.015 0.015 0.01 0.005 0.002 0 0 0 0 0 0];%coupon rate
% risk_free_rate=xlsread('data','rate','C2:C900')';
% rate_T=xlsread('data','rate','E2:E900')';% before interp

%%
[rate_T,index]=unique(rate_T);
risk_free_rate=risk_free_rate(index);%get rid of same value

%%
%---------computation from parameter-------------
k=F/conversion_price;%conversion ratio
ds=Smax/Ns;%price direction
dt=T/Nt;%time direction
rho=1000000/(dt*dt);%penalty parameter

%%
%---------boundary conditions---------
i=(1:Ns+1)';%stock price, [1,2,...,Ns+1],u(x,y)=u(T-xdt,(y-1)dh)
S=(0:Ns)'*ds;%stock price, [0,h,2h,...,Ns*h]
Bc_T=Bc_star+final_coupon_rate*F;%Bc in the final time,using dirty price
Bp_T=Bp_star+final_coupon_rate*F;
```

```
k_T=F/(conversion_price-q*S0*ceil(T-0.5));%dividend influence
u(1:Ns+1,:)=max(k_T*S,F+final_coupon_rate*F);%from t=0 update u to t=T; this is initial value of u when t=0,t=T-t,backward
u=max(min(u,max(Bc_T,k_T*S)),max(Bp_T,k_T*S));%→explict determine value of u by considering constraint
B(1:Ns+1,:)=F+final_coupon_rate*F;%from t=0 update B to t=T; this is initial value of B when t=0,t=T-t,backward
% umax(1:Nt)=k*Smax;%s=Smax,u value, when u_ss=0
% Bmax(1:Nt)=F+final_coupon_rate*F;%s=Smax,B value,B_ss=0;――%this
% two condition is including in the matrix following, so they are useless

%%
%――――――solve pde by difference method, combined penalty method――――――――
count=0;%total iteration time
count1=0;
count2=0;
count3=0;
count4=0;
count5=0;
flag=0;%if not convergenc, break will stop the whole loop

for n=1:Nt%t=T-n*dt,n=1 means t=T-dt,n=N tmeans t=0 %want to u^n→u^(n+1)
    %computation using r(t)
    rate1=interp1(rate_T,risk_free_rate,n*dt-eps,'linear');% discount rate for time step n
    rate2=interp1(rate_T,risk_free_rate,(n-1)*dt-eps,'linear');% discount rate for time step n+1
    r=-(log(rate2/rate1)/dt); %risk free rate in time step n

    alpha=((sigma^2*S(2:Ns,:).^2)/(2*ds^2)-(r+p*eta-q)*S(2:Ns,:)./(2*ds))*dt;%Ns-1, namely alpha_1~alpha_(Ns-1) means ds~(Ns-1)*ds
    beta=((sigma^2*S(2:Ns,:).^2)/(2*ds^2)+(r+p*eta-q)*S(2:Ns,:)./(2*ds))*dt;
    alpha_MuMB=[alpha;0;0];%take Mu and MB down diagonals as alpha_MuMB
    beta_MuMB=[0;0;beta];%take Mu and MB up diagonals as beta_MuMB
    gamma_Mu=[-(r+p)*dt;-(alpha+beta+(r+p)*dt);0];%take Mu diagonals as gamma_Mu
    Mu=spdiags([alpha_MuMB,gamma_Mu,beta_MuMB],(-1:1),Ns+1,Ns+1);%this is Mu
    gamma_MB=[-(r+p*(1-R))*dt;-(alpha+beta+(r+p*(1-R))*dt);0];%take MB diagonals as gamma_MB
    MB=spdiags([alpha_MuMB,gamma_MB,beta_MuMB],(-1:1),Ns+1,Ns+1);%This is MB

    %
    if flag==1
        break;
    end
    t=n*dt;%time to maturaty
    AccI=(ceil(t)-t)*coupon_time_rate(ceil(t))*F;%counpond rate, ACCI
    if (no_call_time(1)<=t)&&(t<=no_call_time(2))
        Bc_n=1000;%when n value of Bc,take a large number means no call
    else
        Bc_n=Bc_star+AccI;%dirty price
    end
    if (no_put_time(1)<=t)&&(t<=no_put_time(2))
        Bp_n=0.01;%small number means no put
    else
        Bp_n=Bp_star+AccI;
    end
```

```
    if (no_convert_time(1)<=t)&&(t<=no_convert_time(2))
        k_n=0.01;%small means no converte
    else
        k_n=F/(conversion_price-q*S0*(ceil(T-0.5)-ceil(t-0.5)));%when T=X.5, stock pay dividends
    end
        u_n=u;%now value of u is n-1 step, will converte to n step following
        B=(eye(Ns+1)-theta*MB)\((eye(Ns+1)+(1-theta)*MB)*B);%similar to eqation (3.45), without constratint, B's eqation,
        % solution B_n is n step iteration initial value
        B(B>Bc_n)=Bc_n;%->explict B<=Bc
        Muu=p*dt*max(k_n*(1-eta)*S,R*B);%similar to (3.44) the right term
        u=(speye(Ns+1)-theta*Mu)\((speye(Ns+1)+(1-theta)*Mu)*u_n+[Muu(1:Ns,:);0]);
        %similar to eqation (3.44), without constratint, u's eqation, solution u_n is n step iteration initial value
        u=max(min(u,max(Bc_n,k_n*S)),max(Bp_n,k_n*S));%->explict constraint u, initial value of u_n
        P1_old=rho*spdiags(u<max(Bp_n,k_n*S),0,Ns+1,Ns+1);
        P2_old=-rho*spdiags(u>max(Bc_n,k_n*S),0,Ns+1,Ns+1);
        % P1,P2 penalty parameter initial value, when this item =0, penalty parameter is 0,
        % otherwise is rho, P1_n^0 is max(Bp_n,kS),P2_n^0 is max(Bc_n,kS)

        while true
        count=count+1;
        u_old=u;%u_old is u^0_n,i.e.,n step initial value
        P1=rho*spdiags(u<max(Bp_n,k_n*S),0,Ns+1,Ns+1);
        P2=-rho*spdiags(u>max(Bc_n,k_n*S),0,Ns+1,Ns+1);
        u=(speye(Ns+1)-theta*Mu+P1-P2)\((speye(Ns+1)+(1-theta)*Mu)*u_n+[Muu(1:Ns,:);0]+P1*max(Bp_n,k_n*S)-P2*max(Bc_n,k_n*S));
```

```
        %similar to (4.28), solution is u_n^(k+1)
        P1=rho*spdiags(u<=max(Bp_n,k_n*S),0,Ns+1,Ns+1);
        P2=-rho*spdiags(u>=max(Bc_n,k_n*S),0,Ns+1,Ns+1);
        if (isequal(P1, P1_old) && isequal(P2, P2_old))
            count1=count1+1;
            break;
        elseif max(abs(u-u_old)./max(1,abs(u)))<1/sqrt(rho)
            count2=count2+1;
                break;
            elseif count>Nt*40
                    count4=count4+1;
                    disp(['Probably does not converge, current T is ' num2str(T)]);
                    flag=1;
                    break;
        else
            count3=count3+1;
        end
        P1_old=P1;
        P2_old=P2;%fix n,every interation of k, u,P1,P2 will change value, so add 'old'
        end
        if (ceil(t+dt)-ceil(t))==1%coupon rate payment
            count5=count5+1;
            u=u+coupon_time_rate(ceil(t+dt+1e-9))*F;
            B=B+coupon_time_rate(ceil(t+dt+1e-9))*F;
```

```
        end
end

%%
U=u(ceil(Ns*S0/Smax));
% usingtime=toc;
% Out=['The convertible value is ', num2str(U)];
% disp(Out);
% disp(['Total elapsed time : ' num2str(usingtime) ', Nt = ' num2str(Nt) ', Ns = ' num2str(Ns)]);
end
```

We use time-dependent risk free rate $r(t)$, to be detail, when calculating the price at t_0, we use linear interpolation to get a risk free rate curve at t_0, then for every time grid t, we use the forward risk free rate from $t+1$ to t as the risk free rate. As mentioned, we use penalty method to solve the PDE, and using backward fully implicit method to calculate from time $t+1$ to t, and using Newton iteration to get the solution of equation.

4.1.2 Calculating B_c^* and B_p^*

The following code is calculating the equivalent trigger price B_c^* and B_p^*.

```
function pdifference = calculateBcstar(S0, Bc, Bc_star,T,risk_free_rate,rate_T,sigma,n,m)
%% parameter input
% clc;
% clear;
% S0=18.21;
% Bc=43.05*1.3;
% Bc_star=62.2146;
% T=4.08;
% u=-0.01;
% sigma=0.42;
% n=15;
% m=30;

N=10000;%numbers of Mc
d_t=1/252;
M=ceil(T/d_t);
% r=xlsread('data','rate','C2:C900)';
% rate_T=xlsread('data','rate','E2:E900)';% before interp
[rate_T,index]=unique(rate_T);
risk_free_rate=risk_free_rate(index);%get rid of same value
%% calculate
left_sum=zeros(1,N);
right_sum=zeros(1,N);
windowSize=m;
filter_b=(1/windowSize)*ones(1,windowSize);%m moving average
```

```
filter_a=1;%weight
for i=0:M
    rate1=interp1(rate_T,risk_free_rate,i*d_t-eps,'linear');% discount rate for time step i
    rate2=interp1(rate_T,risk_free_rate,(i-1)*d_t-eps,'linear');% discount rate for time step i
    r(i+1)=-(log(rate2/rate1)/d_t); %risk free rate in time step i
end
for i=1:N
    W=[0,cumsum(random('norm',0,sqrt(d_t),1,M))];
    S=S0.*exp((r-sigma^2/2).*(0:M).*d_t+sigma*W);
    judge=(S>Bc);
    judge_filter=filter(filter_b,filter_a,judge);
    if any (judge_filter>=n/m-eps)
        left_sum(i)=1;
    end
    judgestar=(S>Bc_star);
    if any(judgestar==1)
        right_sum(i)=1;
    end
end
pbc=mean(left_sum);
pbcstar=mean(right_sum);
pdifference=pbc-pbcstar;
end
```

```
function pdifference = calculateBstar(S0, Bp, Bp_star,T,risk_free_rate,rate_T,sigma,n,m)
%% parameter input
% clc;
% clear;
% S0=38.79;
% Bp=18.501;
% Bp_star=1.85;
% T=5.5472;
% u=0.0093;
% sigma=3.1551;
% n=30;
% m=30;

N=10000;%numbers of Mc
d_t=1/252;
M=ceil(T/d_t);
%risk_free_rate=xlsread('data','rate','C2:C900')';
%rate_T=xlsread('data','rate','E2:E900')';% before interp
[rate_T,index]=unique(rate_T);
risk_free_rate=risk_free_rate(index);%get rid of same value

%% calculate
left_sum=zeros(1,N);
right_sum=zeros(1,N);
windowSize=m;
```

```
filter_b=(1/windowSize)*ones(1,windowSize);%m moving average
filter_a=1;%weight
for i=0:M
    rate1=interp1(rate_T,risk_free_rate,i*d_t-eps,'linear');% discount rate for time step i
    rate2=interp1(rate_T,risk_free_rate,(i-1)*d_t-eps,'linear');% discount rate for time step i
    r(i+1)=-(log(rate2/rate1)/d_t); %risk free rate in time step i
end
for i=1:N
    W=[0,cumsum(random('norm',0,sqrt(d_t),1,M))];
    S=S0.*exp((r-sigma^2/2).*(0:M)*d_t+sigma*W);
    judge=(S<Bp);
    judge_filter=filter(filter_b,filter_a,judge);
    if any (judge_filter>=n/m-eps)
        left_sum(i)=1;
    end
    judgestar=(S<Bp_star);
    if any(judgestar==1)
        right_sum(i)=1;
    end
end
pbp=mean(left_sum);
pbpstar=mean(right_sum);
pdifference=pbp-pbpstar;
end
```

We also use time dependent risk free rate, and take MC times $N = 10000$, we take time step length as one day, for the call feature, it is 15 out of 30 days, and the put feature is 30 out of 30 days.

4.1.3 An One Day Example

In this part we give an one day example to explain how we calculate the price. We have all information for $2017-6-9$, and also we have know the stock price and risk free rate curve for $2017-6-10$, using these information, we calculate the convertible price for day $2017-6-10$.

```
%%
clc;
clear;
tic;
%%
%read data
ConversionPrice=xlsread('data','ge_er','E597:E598')';% yesterday and today convertion price
MarketPrice=xlsread('data','ge_er','D597')';% yesterday market price
S=xlsread('data','ge_er','H597:H598')';% yesterday and today stock price
T=xlsread('data','ge_er','C597:C598')'; % yesterday and today time to maturaty
risk_free_rate=xlsread('data','rate','C2:C900')';% discount_rate
rate_T=xlsread('data','rate','E2:E900')';% before interp

%set up parameter
Nt=400;
Ns=200;
FaceValue=100;
Smax=100;
FinalCouponRate=0.08;
NoConverTime=[5.38 6];
NoCallTime=[5.38 6];
NoPutTime=[3.88 6];
Bc=1.3*100;
Bp=0.7*100;
theta=1;
```

```
ConvertValue=S.*(100./ConversionPrice);%转股价值
coupon_time_rate=[0.016 0.016 0.016 0.01 0.007 0.005 0 0 0 0 0 0];
q=0;%dividend
p=0;%default probobality
R=0.5;%recovery rate
eta=0.5;
n1=15;%bc
n2=30;%bp
m=30;

%% Calculation of Bc_star and Bp_star and implied volatility

Bc_stock=Bc./(FaceValue./ConversionPrice);% Bc boundary---stock price
Bp_stock=Bp./(FaceValue./ConversionPrice);
implied_vol=0.4;% iv initial value of yesterday
for j=1:5 %iteration time, calculate yesterday equ bcstar, equ bpstar and implied vol
    % Bc interation
    j
    Bcstar_fun=@(Bcstar) calculateBcstar(S(1),Bc_stock(1),Bcstar,T(1),risk_free_rate,rate_T,max(min(implied_vol,5),0.05),n1,m);
    Bc_stock0=[0.5*S(1),10*Bc_stock(1)];% initial value of Bcstar
    options=optimset('Display','iter','TolX',0.01);
    Bc_stock_star=fzero(Bcstar_fun,Bc_stock0,options);
    Bc_star=Bc_stock_star*(FaceValue/ConversionPrice(1)); % equivalence Bc boundary of yesterday
    % Bp iteration
```

```
    Bpstar_fun=@(Bpstar) calculateBpstar(S(1),Bp_stock(1),Bpstar,T(1),risk_free_rate,rate_T,max(min(implied_vol,5),0.05),n2,m);
    Bp_stock0=[0.1*Bp_stock(1),2*S(1)];%initial value of Bpstar
    options=optimset('Display','iter','TolX',0.01);
    Bp_stock_star=fzero(Bpstar_fun,Bp_stock0,options);
    Bp_star=Bp_stock_star*(FaceValue/ConversionPrice(1));% equivalence Bp boundary of yesterday
    % implied vol iteration
    iv0=[0.05,5];% initial value of Bcstar
    if (convertible_code(Nt,Ns,FaceValue,ConversionPrice(1),5*S(1),S(1),risk_free_rate,rate_T,5,...
        T(1),FinalCouponRate,q,p,R,eta,NoCallTime,NoPutTime,NoConverTime,Bc_star,Bp_star,theta,coupon_time_rate)-MarketPrice>0)...
    &&(convertible_code(Nt,Ns,FaceValue,ConversionPrice(1),5*S(1),S(1),risk_free_rate,rate_T,0.05,...
        T(1),FinalCouponRate,q,p,R,eta,NoCallTime,NoPutTime,NoConverTime,Bc_star,Bp_star,theta,coupon_time_rate)-MarketPrice<0)
    % this is judge that iv is not beyond the iterval [0.05, 0.5], otherwise we take its value as 0.05 or 0.5

    iv_fun=@(sigma) convertible_code(Nt,Ns,FaceValue,ConversionPrice(1),5*S(1),S(1),risk_free_rate,rate_T,sigma,...
        T(1),FinalCouponRate,q,p,R,eta,NoCallTime,NoPutTime,NoConverTime,Bc_star,Bp_star,theta,coupon_time_rate)-MarketPrice;
    options=optimset('Display','iter','TolX',0.001);
    implied_vol=fzero(iv_fun,iv0,options); % get value of implied vol
    elseif (convertible_code(Nt,Ns,FaceValue,ConversionPrice(1),5*S(1),S(1),risk_free_rate,rate_T,5,...
        T(1),FinalCouponRate,q,p,R,eta,NoCallTime,NoPutTime,NoConverTime,Bc_star,Bp_star,theta,coupon_time_rate)-MarketPrice<=0)
        implied_vol=5;
    else
        implied_vol=0.05;
    end
end
```

```
%% Main program
TodayPrice=convertible_code(Nt,Ns,FaceValue,ConversionPrice(2),5*S(2),S(2),risk_free_rate,rate_T,implied_vol,...
    T(2),FinalCouponRate,q,p,R,eta,NoCallTime,NoPutTime,NoConverTime,Bc_star,Bp_star,theta,coupon_time_rate);
% using yesterday's implied vol and Bcstar, Bpstar
disp(['Today convertible bond theory price is ' num2str(TodayPrice) ',time= ' num2str(toc)]);
```

As the code shown, we assume $q = 0$, namely there is no dividend, and the default probability is $p = 0$, original B_c is 130, B_p is 70. As the volatility input, we use implied volatility of $2017-6-9$ and equivalent boundary B_c^*, B_p^* of $2017-6-9$ as volatility input of $2017-6-10$, and equivalent boundary input of $2017-6-10$. Since to calculate B_c^* and B_p^* is also requested for volatility input, we use an iteration method to get the implied volatility and equivalent boundary of day $2017-6-9$: we first use an initial volatility 0.4 as volatility input to calculate B_c^* and B_p^*, than get value of B_c^* and B_p^*, and use this equivalent boundary as equivalent boundary input to calculate a price such that this price equals the market price of day $2017-6-9$. Thus get the implied volatility and use take this implied volatility as the new input of B_c^* and B_p^*, and so on, until the implied volatility and equivalent don't change a lot. If the implied volatility is more than 5 or less than 0.05. we take its value as 5 or 0.05.

4.2 Some Numerical Results

The following are two numerical results. We implement to convertible bonds, Geer and Guangqi. Guangqi: The time is from 2016/2/26 to 2017/3/7, from year 0.09 to year 1.12. We consider the call and put provision, and use a new B_c^* B_p^* to replace the old one. The call feature is 15-out-of-30, the trigger price is 130% of convertible price. The put feature is 30-out-of-30, the trigger price is 70% of convertible price. The convertible-period and call-period is from year 0.5 to year 6, namely after 2016/7/21. The put-period is from year 4 to year 6. The stock dividend is paid once a year, we take $q = 0$ now. The default risk $p = 0$. The pay of interest rate is considered, it is assumed coupon interest rate is paid once a year, at $T = 5, 4, 3, 2, 1, 0$, so we use dirty price.

Geer is similar to Guangqi except the time is from 2015/1/14 to 2017/3/7, from year 0.09 to year 2.24. The convertible-period and call-period is from year 0.5 to year 6, namely after 2015/6/11. The put-period is from year 2 to year 6.

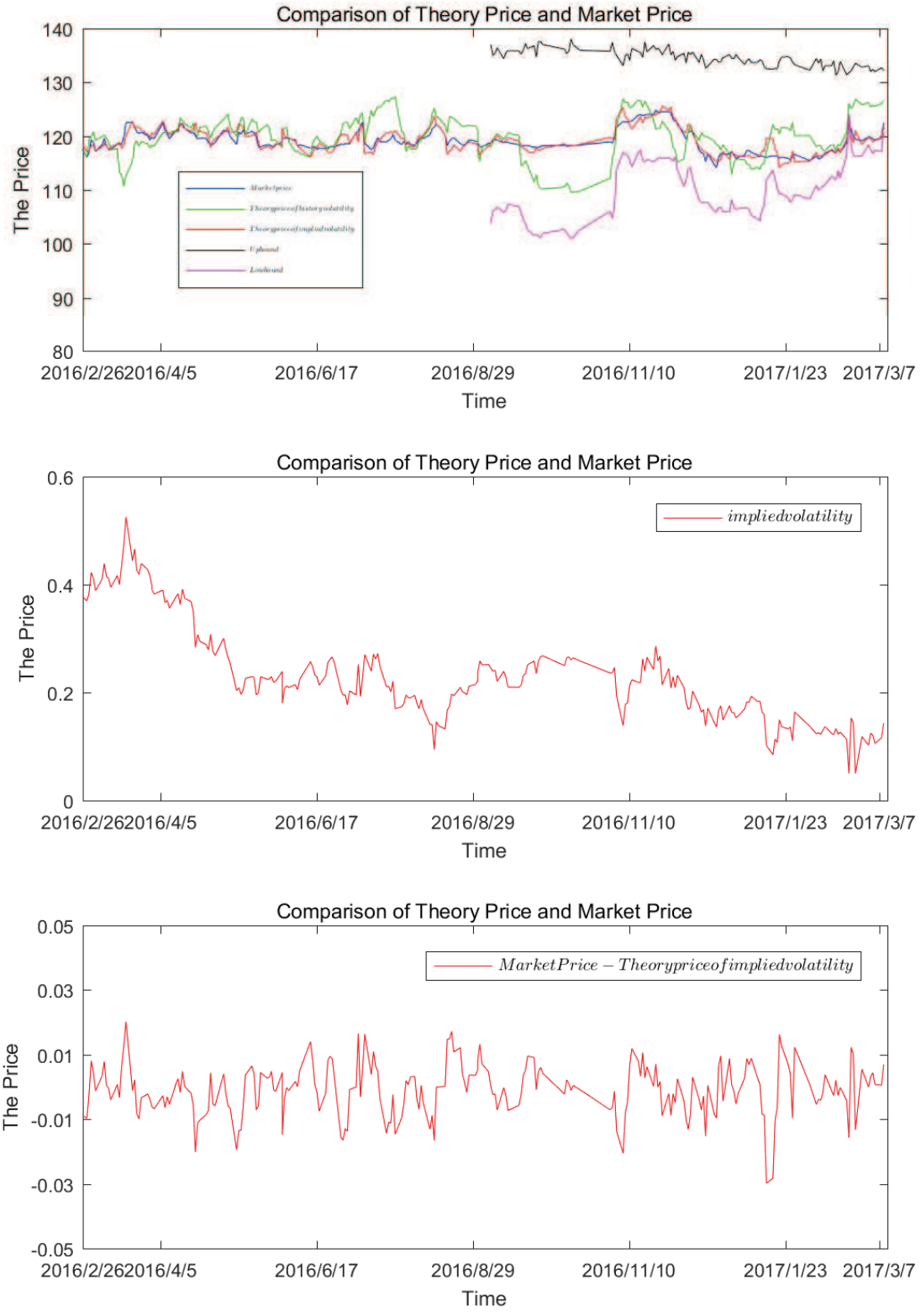
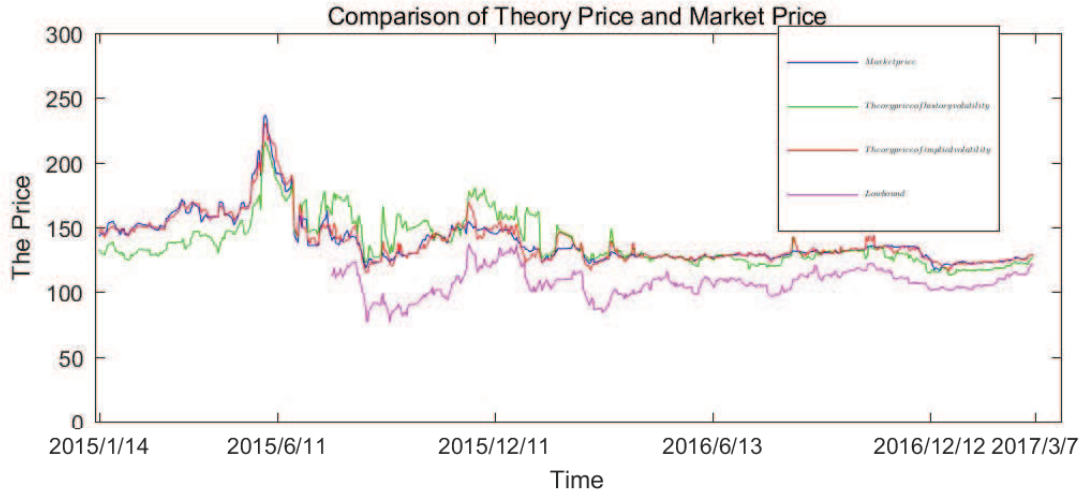
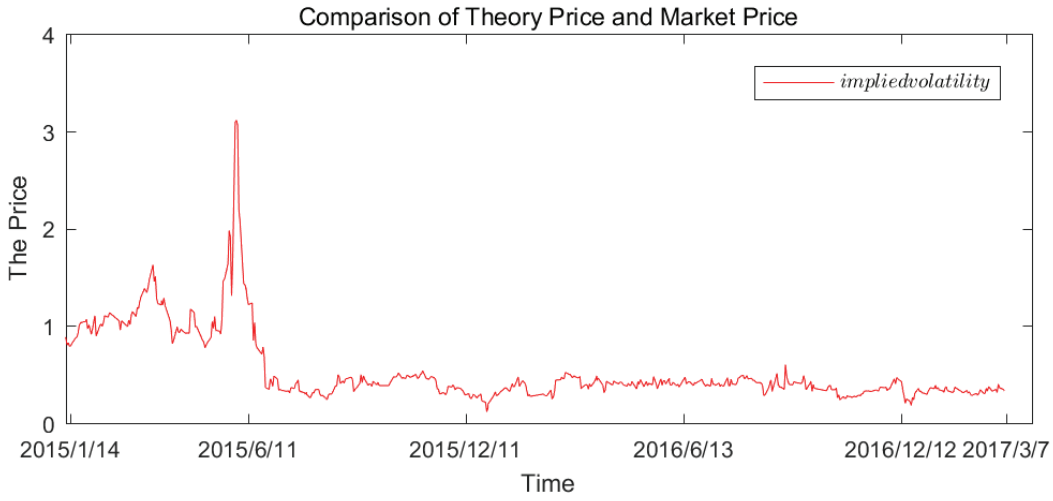


Figure 1: The price of Guangqi with implied volatility by PDE method





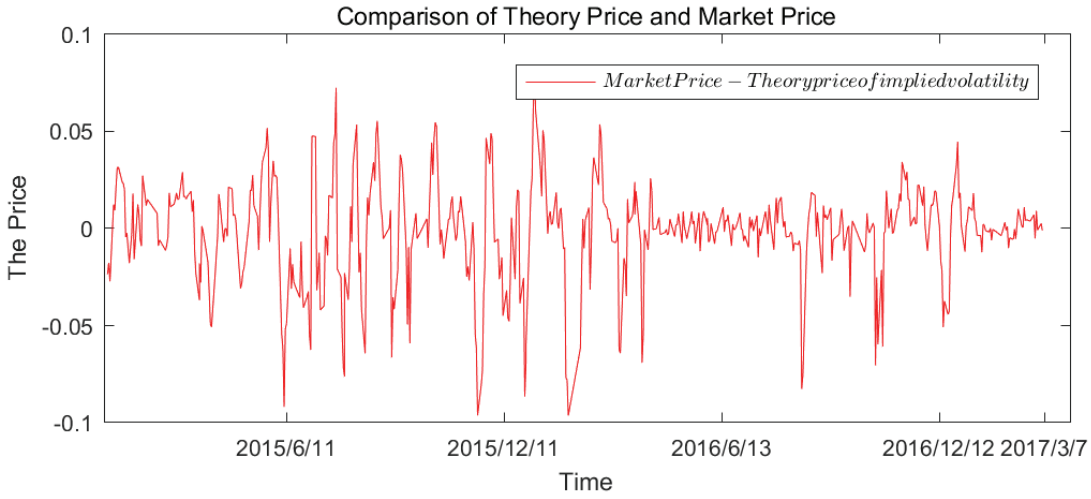


Figure 2: The price of Geer with implied volatility by PDE method

References

[1] Li L X. Pricing Convertible Bonds using Partial DiFFerential Equations[D]. University of Toronto, 2006.

[2] Ayache E, Forsyth P A, Vetzal K R. Valuation of convertible bonds with credit risk[J]. The journal of Derivatives, 2003, 11(1): 9-29.

[3] Zhang J X. Back to the Future: An Approximate Solution for N Out of M Soft-Call Option[J]. 2012.