Valuation of Contingent Convertibles with Derivatives

Nicolay Schmitt

Frankfurt School of Finance and Management

A thesis submitted for the degree of

Master of Science in Finance

August 2016

This thesis is dedicated to
my parents
for their love and support.
Thank you!

Acknowledgements

plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle.

Abstract

plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle, plenty of waffle.

Contents

List of Figures

Chapter 1

Introduction and Motivation

1.1 Introduction

1.2 Literature Overview

1.3 Motivation

1.4 Methodology

Chapter 2

Structure of CoCos

2.1 Description of CoCos

2.2 Payoff and Risk Profile

2.3 Conversion Trigger

2.3.1 Market Trigger

2.3.2 Accounting Trigger

2.3.3 Regulatory Trigger

2.3.4 Multivariate Trigger

2.4 Conversion Details

2.4.1 Conversion Fraction

- conversion fraction α
- face value N
- conversion amount $N \times \alpha$
- amount remaining in case of partial equity conversion $N \times (1 - \alpha)$

2.4.2 Conversion Price and Ratio

- conversion rate C_r
- conversion price C_p

- recovery rate R_{CoCo}
- stock price at trigger event S_T^*
- loss attributable to CoCo holders L_{CoCo}

$$C_p = \frac{\alpha N}{C_r} \tag{2.1}$$

$$C_r = \frac{\alpha N}{C_p} \tag{2.2}$$

$$R_{CoCo} = \frac{S_T^*}{C_p} \tag{2.3}$$

$$L_{CoCo} = N - (1 - R_{CoCo}) = N\left(1 - \frac{S_T^*}{C_p}\right) \tag{2.4}$$

$$P_T = \begin{cases} (1-\alpha)N + C_r S_T^* & \text{if converted} \\ N & \text{if not converted} \end{cases} \tag{2.5}$$

Chapter 3

Theory of Pricing

3.1 Credit Derivative Approach

3.1.1 Intensity-based Approach

$$p^* = 1 - \exp\left(-\lambda_{Trigger} \times T\right) \tag{3.1}$$

$$s_{CoCo} = (1 - R_{CoCo}) \times \lambda_{Trigger} = Loss_{CoCo} \times \lambda_{Trigger} \tag{3.2}$$

$$Loss_{CoCo} = N - C_r \times S^* = N\left(1 - \frac{S^*}{C_P}\right) \tag{3.3}$$

$$R_{CoCo} = \frac{S^*}{C_p} \tag{3.4}$$

$$p^* = \Phi\left(\frac{\log\left(\frac{S^*}{S}\right) - \mu T}{\sigma\sqrt{T}}\right) + \left(\frac{S^*}{S}\right)^{\frac{2\mu}{\sigma^2}} \Phi\left(\frac{\log\left(\frac{S^*}{S}\right) + \mu T}{\sigma\sqrt{T}}\right) \tag{3.5}$$

$$\lambda_{Trigger} = -\frac{\log\left(1 - p^*\right)}{T} \tag{3.6}$$

$$s_{CoCo} = -\frac{\log\left(1 - p^*\right)}{T} \times \left(1 - \frac{S^*}{C_p}\right) \tag{3.7}$$

3.1.2 Application to CoCos

3.1.3 Data Requirements and Calibration

3.1.4 Pricing Example

3.2 Equity Derivative Approach

Sources: [?], [?]

$$\begin{aligned} P_T &= \mathbb{1}_{\{\tau>T\}}N + \left[(1-\alpha)\,N + \frac{\alpha N}{C_p S^*}\right]\mathbb{1}_{\{\tau\leq T\}} \\ &= N + \left[\frac{\alpha N}{C_p}S^* - \alpha N\right]\mathbb{1}_{\{\tau\leq T\}} \\ &= N + \left[C_r S^* - \alpha N\right]\mathbb{1}_{\{\tau\leq T\}} \\ &= N + C_r\left[S^* - \frac{\alpha N}{C_r}\right]\mathbb{1}_{\{\tau\leq T\}} \\ &= N + C_r\left[S^* - C_p\right]\mathbb{1}_{\{\tau\leq T\}} \end{aligned}$$

$$V_t^{ed} = V_t^{cb} - V_{t_i}^{dibi} + V_t^{difwd} \tag{3.8}$$

3.2.1 Corporate Bonds

$$V_t^{cb} = \sum_{i=t}^{T} c_i \exp\left(-rt_i\right) + N\exp\left[-r\left(T-t\right)\right] \tag{3.9}$$

3.2.2 Binary Options

$$\begin{aligned} V_t^{dibi}\left(c_i, S^*, t\right) = \alpha\sum_{i=1}^{k} c_i \exp\left(-rt_i\right)\Bigg[&\Phi\left(-x_{1i} + \sigma\sqrt{t_i}\right) \\ &+ \left(\frac{S^*}{S_t}\right)^{2\lambda-2}\Phi\left(y_{1i} - \sigma\sqrt{t_i}\right)\Bigg] \end{aligned} \tag{3.10}$$

with

$$x_{1i} = \frac{\log\left(\frac{S_t}{S^*}\right)}{\sigma\sqrt{t_i}} + \lambda\sigma\sqrt{t_i}$$
$$y_{1i} = \frac{\log\left(\frac{S^*}{S_t}\right)}{\sigma\sqrt{t_i}} + \lambda\sigma\sqrt{t_i}$$
$$\lambda = \frac{r - q + \frac{\sigma^2}{2}}{\sigma^2}$$

3.2.3 Down-And-In Forward

$$\max\left(S_T - K\right) \text{ if } \min_{0\leq t\leq T}\left(S_T\right) \leq S^* \tag{3.11}$$

$$\max\left(K - S_T\right) \text{ if } \min_{0\leq t\leq T}\left(S_T\right) \leq S^* \tag{3.12}$$

$$\begin{aligned} V_t^{dic}\left(S_t, S^*, K\right) = {} & S_t \exp\left[-q\left(T-t\right)\right]\left(\frac{S^*}{S_t}\right)^{2\lambda}\Phi\left(y\right) \\ & - K\exp\left[-r\left(T-t\right)\right]\left(\frac{S^*}{S_t}\right)^{2\lambda-2}\Phi\left(y - \sigma\sqrt{T-t}\right) \end{aligned} \tag{3.13}$$

with

$$K = C_p$$
$$y = \frac{\log\left(\frac{S^{*2}}{S_t K}\right)}{\sigma\sqrt{T-t}} + \lambda\sigma\sqrt{T-t}$$
$$\lambda = \frac{r - q + \frac{\sigma^2}{2}}{\sigma^2}$$

$$\begin{aligned} V_t^{dip}\left(S_t, S^*, K\right) = {} & S_t \exp\left[-q\left(T-t\right)\right]\left(\frac{S^*}{S_t}\right)^{2\lambda}\left[\Phi\left(y\right) - \Phi\left(y_1\right)\right] \\ & - K\exp\left[-r\left(T-t\right)\right]\left(\frac{S^*}{S_t}\right)^{2\lambda-2}\left[\Phi\left(y - \sigma\sqrt{T-t}\right) - \Phi\left(y_1 - \sigma\sqrt{T}\right)\right] \\ & + K\exp\left[-r\left(T-t\right)\right]\Phi\left(x_1 + \sigma\sqrt{T-t}\right) \\ & - S_t\exp\left[-q\left(T-t\right)\right]\Phi\left(-x_1\right) \end{aligned} \tag{3.14}$$

with

$$x_1 = \frac{\log\left(\frac{S_t}{S^*}\right)}{\sigma\sqrt{T-t}} + \lambda\sigma\sqrt{T-t}$$
$$y_1 = \frac{\log\left(\frac{S^*}{S_t}\right)}{\sigma\sqrt{T-t}} + \lambda\sigma\sqrt{T-t}$$

$$\min(S_t) \leq S^* : P_T = S_T - K = \max(S_T - K) - \max(K - S_T) \tag{3.15}$$

$$\min(S_t) > S^* : P_T = 0 \tag{3.16}$$

$$\begin{aligned} V_t^{difwd} = C_r \Bigg[& S_t \exp\left[-q\left(T-t\right)\right] \left(\frac{S^*}{S_t}\right)^{2\lambda} \Phi\left(y_1\right) \\ & - K \exp\left[-r\left(T-t\right)\right] \left(\frac{S^*}{S_t}\right)^{2\lambda-2} \Phi\left(y_1 - \sigma\sqrt{T-t}\right) \\ & - K \exp\left[-r\left(T-t\right)\right] \Phi\left(-x_1 - \sigma\sqrt{T-t}\right) \\ & + S_t \exp\left[-q\left(T-t\right)\right] \Phi\left(-x_1\right) \Bigg] \end{aligned} \tag{3.17}$$

with

$$C_r = \frac{\alpha N}{C_p} \tag{3.18}$$

3.2.4 Data Requirements and Calibration

3.2.5 Pricing Example

Chapter 4

Dynamics and Sensitivity Analysis

4.1 Credit Derivative Approach

4.2 Equity Derivative Approach

Chapter 5

Empirical Analysis and Model Comparison

5.1 Data Description

5.1.1 Deutsche Bank

5.2 Model Parametrization

5.3 Model Comparison

5.3.1 Qualitative Analysis

5.3.2 Quantitative Analysis

Chapter 6

Conclusion

Appendix A

Sample Title

Appendix B

Code - Credit Derivative Approach

The following source code is an implementation of the Credit Derivative Approach written in R.

```
# Price of Contingent Convertible Bond
price_coco_cd <- function(t, T, S_t, S_star, C_p, c_i, r, N, q, sigma){

  spread_coco <- calc_spread_coco(t, T, S_t, S_star, C_p, r, q, sigma)
  V_t_coco <- N * exp(-(r + spread_coco) * (T - t))

  for (t in 1:T){
    V_t_coco <- V_t_coco + c_i * exp(-(r + spread_coco) * t)
  }
  V_t_coco
}

# Calculation of Trigger Probability
calc_p_star <- function(t, T, S_t, S_star, r, q, sigma){
  p_star <- pnorm((log(S_star / S_t) - calc_mu(r, q, sigma) * (T - t)) /
    (sigma * sqrt(T - t))) + (S_star / S_t)^(2 * calc_mu(r, q, sigma) /
    sigma^2) * pnorm((log(S_star / S_t) + calc_mu(r, q, sigma) * (T - t
   )) / (sigma * sqrt(T - t)))
  p_star
}

# Calculation of Drift of Underlying
calc_mu <- function(r, q, sigma){
  mu <- r - q - sigma^2 / 2
  mu
}

# Spread of CoCo Bond
calc_spread_coco <- function(t, T, S_t, S_star, C_p, r, q, sigma){
  spread_coco <- - log(1 - calc_p_star(t, T, S_t, S_star, r, q, sigma))
   / (T - t) * (1 - S_star / C_p)
  spread_coco
}

# Pricing Example
```

```
price_coco_cd(t <- 0, T <- 5, S_t <- 40, S_star <- 20, C_p <- 25, c_i <-
    7, r <- 0.03, N <- 100, q <- 0, sigma <- 0.3)
```

Appendix C

Code - Equity Derivative Approach

The following source code is an implementation of the Equity Derivative Approach written in R.

```
# Price of Contingent Convertible Bond
price_coco_ed <- function(t, T, S_t, S_star, C_p, c_i, r, N, q, sigma, alpha){
  V_t_ed <- price_cb(t, T, c_i, r, N) - price_dibi(t, T, S_t, S_star, c_i, r, q, sigma, alpha) + price_difwd(t, T, S_t, S_star, C_p, r, N, q, sigma, alpha)

  return(V_t_ed)
}

# Price of Corporate Bond
price_cb <- function(t, T, c_i, r, N){
  V_t_cb <- N * exp(-r * (T - t))

  for (t in 1:T){
  V_t_cb <- V_t_cb + c_i * exp(-r * t)
  }

  return(V_t_cb)
}

# Price of Binary Option
price_dibi <- function(t, T, S_t, S_star, c_i, r, q, sigma, alpha){
  V_t_dibi <- 0

  i <- t
  k <- T

  for (i in 1:k) {
  V_t_dibi <- V_t_dibi + c_i * exp(- r * i) * (pnorm(- calc_x_1_i(S_t, S_star, sigma, r, q, i) + sigma * sqrt(i)) + (S_star / S_t)^(2 * calc_lambda(r, q, sigma) - 2) * pnorm ( calc_y_1_i(S_t, S_star, sigma, r, q, i) - sigma * sqrt(i)))
  }

```

```
  V_t_dibi <- alpha * V_t_dibi

  return(V_t_dibi)
}

# Price of Down-And-In Forward
price_difwd <- function(t, T, S_t, S_star, C_p, r, N, q, sigma, alpha){
  V_t_difwd <- calc_conversion_rate(C_p, N, alpha) * (S_t * exp(- q * (T
    - t)) * (S_star / S_t) ^ (2 * calc_lambda(r, q, sigma)) * pnorm(
    calc_y_1(t, T, S_t, S_star, r, q, sigma)) - C_p * exp(- r * (T - t))
     * (S_star / S_t)^(2 * calc_lambda(r, q, sigma) - 2) * pnorm(calc_y_
    1(t, T, S_t, S_star, r, q, sigma) - sigma * sqrt(T - t)) - C_p * exp
    (- r * (T - t)) * pnorm(- calc_x_1(t, T, S_t, S_star, r, q, sigma) +
     sigma * sqrt(T - t)) + S_t * exp(- q * (T - t)) * pnorm(- calc_x_1(
    t, T, S_t, S_star, r, q, sigma)))

  return(V_t_difwd)
}

# Calculation of Conversion Rate
calc_conversion_rate <- function(C_p, N, alpha){
  C_r <- alpha * N / C_p

  return(C_r)
}

# Calculation of additional Parameters
calc_x_1_i <- function(S_t, S_star, sigma, r, q, t_i){
  x_1_i <- log(S_t / S_star) / (sigma * sqrt(t_i)) + calc_lambda(r, q,
    sigma) * sigma * sqrt(t_i)

  return(x_1_i)
}

calc_y_1_i <- function(S_t, S_star, sigma, r, q, t_i){
  y_1_i <- log(S_star / S_t) / (sigma * sqrt(t_i)) + calc_lambda(r, q,
    sigma) * sigma * sqrt(t_i)

  return(y_1_i)
}

calc_lambda <- function(r, q, sigma){
  lambda <- (r - q + sigma^2 / 2) / sigma^2

  return(lambda)
}

calc_x_1 <- function(t, T, S_t, S_star, r, q, sigma){
  x_1 <- log(S_t / S_star) / (sigma * sqrt(T - t)) + calc_lambda(r, q,
    sigma) * sigma * sqrt(T - t)

  return(x_1)
}

```

```
74 calc_y_1 <- function(t, T, S_t, S_star, r, q, sigma){
75   y_1 <- log(S_star / S_t) / (sigma * sqrt(T - t)) + calc_lambda(r, q,
       sigma) * sigma * sqrt(T - t)
76
77   return(y_1)
78 }
79
80 # Pricing Example
81 price_coco_ed(t <- 0, T <- 5, S_t <- 40, S_star <- 20, C_p <- 25, c_i <-
       7, r <- 0.03, N <- 100, q <- 0, sigma <- 0.3, alpha <- 1)
```

Appendix D

Code - Structural Approach

The following source code is an implementation of the Structural Approach written in R.

```
price_coco_sa <- function(T , npath , rho , kappa , r.bar = 0.069, r0 =
    0.035, sigma_r = 0.07){
  n <- T * 250
  dt <- T / n

  result <- sim_corrProcess(T, npath, rho, n, dt)
  dW_1 <- result$dW_1
  dW_2corr <- result$dW_2corr

  r <- sim_interestrate(kappa, r.bar, r0, sigma_r, dW_2corr, n, npath,
    dt)

  V_t_sa <- get_price(npath, n, dt, dW_1, dW_2corr, r) * 100

  print(V_t_sa)
}

sim_corrProcess <- function(T, npath, rho, n, dt){
  vect <- c(1, rho, rho, 1)
  RHO <- matrix(vect, nrow = 2)
  chol.RHO <- t(chol(RHO)) ### What does this function do?

  # Create two Brownian Motions
  dW_1 <- matrix(1, n, npath)
  dW_2 <- matrix(1, n, npath)

  for(j in 1:npath)
  {
    dW_1[ , j] <- rnorm(n) * sqrt(dt)
    dW_2[ , j] <- rnorm(n) * sqrt(dt)
  }

  # Create Correlated Process based on Brownian Motions using Cholesky-
    Decomposition
  dW_2corr <- matrix(1, n, npath)
```

```
  for(j in 1:npath)
  {
    for(i in 1:n)
    {
      dW_2corr[i, j] <- dW_1[i, j] * chol.RHO[2, 1] + dW_2[i, j] * chol.
    RHO[2, 2] ### Why are those GBMs correlated?
    }
  }
  return(list("dW_1" = dW_1, "dW_2corr" = dW_2corr))
}

# Create Interest Rate Process
sim_interestrate <- function(kappa, r.bar, r0, sigma_r, dW_2corr, n,
    npath, dt){
  r <- matrix(r0, n + 1, npath)

  for(j in 1:npath)
  {
    for(i in 1:n)
    {
      r[i + 1, j] <- r[i, j] + kappa * (r.bar - r[i, j]) * dt + sigma_r
    * sqrt(r[i, j]) * dW_2corr[i, j] ### Why are those GBMs correlated?
    }
  }
  return(r)
}


get_price <- function(npath, n, dt, dW_1, dW_2corr, r){

  # Initiliaze jump parameters
  mu_Y <- - 0.01
  sigma_Y <- 0.02
  lambda <- c(1, 0, 1)

  #x
  g <- c(0.5, 0.5, 0.25)
  x.hat <- 1.1 # Capital ratio target
  b0 <- 0.04 # CCB-to-deposit ratio, B/D
  p <- 1 # Conversion parameter: par=1, premium>1, discount<1
  e.bar <- 0.02 # equity-to-deposit ratio
  sigma_x <- 0.02 # Standard deviation for assets
  #x0 <- 1.075 # Starting capital-to-deposit ratio
  x0.low <- 1 + 0.065
  x0.high <- 1 + 0.15
  x0.nint <- 10 # number of intervals between x0.low and x0.high
  B <- 1

  c.low <- 0.04
  c.high <- 0.058
  c.nint <- 10
  c.fit.matrix <- matrix(0, x0.nint, length(lambda))


```

```
84  for(w in 1:length(lambda))
85  {
86    # Create parametres for jump process
87    phi <- matrix(rbinom( n%*%npath, 1, dt * lambda[w]), n, npath)
88    ln.Y <- matrix(rnorm(n%*%npath, mu_Y, sigma_Y), n, npath)
89
90    b <- matrix(b0, n + 1, npath) # Ratio of the contingent capital's
      par value to the date t value of deposits
91    x.bar0 <- 1 + e.bar + p * b0
92    x.bar <- matrix(x.bar0, n + 1, npath) # Asset-to-deposit threshold
93
94    h <- matrix(1, n, npath) # Fair deposit insurance premium or deposit
       credit risk premium
95
96    k <- exp(mu_Y + 0.5 * sigma_Y^2) - 1
97
98    c <- seq(c.low, c.high, length = c.nint)
99    x0 <- seq(x0.low, x0.high, length = x0.nint)
100   reg.matrix <- matrix(0, c.nint, length(x0))
101
102   reg.fit1 <- 1:x0.nint
103   reg.fit2 <- 1:x0.nint
104   c.fit <- 1:x0.nint
105
106   for(l in 1:x0.nint)
107   {
108     for(m in 1:c.nint)
109     {
110       x <- matrix(x0[l],n+1,npath)
111       ln.x0 <- matrix(log(x0[l]),n+1,npath)
112       ln.x <- ln.x0
113       binom.c <- matrix(1,n+1,npath)
114
115       for(j in 1:npath)
116       {
117         for(i in 1:n)
118         {
119           d_1 <- (ln.x[i, j] + mu_Y) / sigma_Y
120           d_2 <- d_1 + sigma_Y
121
122           h[i, j] <- lambda[w] * (pnorm( - d_1) - exp(ln.x[i, j]) *
    exp(mu_Y + 0.5 * sigma_Y^2) * pnorm(-d_2))
123
124           b[i + 1, j] <- b[i, j] * exp(- g[w] * (exp(ln.x[i, j]) - x.
    hat) * dt)
125
126           ln.x[i + 1, j] <- ln.x[i, j] + ( (r[i, j] - lambda[w] * k) -
     (r[i, j] + h[i, j] + c[m] * b[i, j]) / exp(ln.x[i, j]) - g[w] * (
    exp(ln.x[i, j]) - x.hat) - 0.5 * sigma_x^2) * dt + sigma_x * sqrt(dt
    ) *dW_1[i,j] + ln.Y[i,j]*phi[i,j]
127
128           x[i + 1, j] <- exp(ln.x[i + 1, j])
129
130           x.bar[i + 1, j] <- 1 + e.bar + p * b[i + 1, j]
```

```
131
132           if(x[i + 1, j] >= x.bar[i + 1, j] && binom.c[i, j] > 0.5)
133           {
134             binom.c[i + 1, j] <- 1
135           }else
136           {
137             binom.c[i + 1, j] <- 0
138           }
139         }
140       }
141
142       payments <- matrix(c(rep(c[m] * dt, n - 1), B), n, npath) *
    binom.c[1:n, ]
143       for(j in 1:npath){
144         for(i in 2:n){
145           if(payments[i, j] == 0 && p * b[sum(binom.c[ , j]) + 1, j]
    <= x[sum(binom.c[ , j]) + 1, j] - 1 ){
146             payments[i, j] <- p * B
147             break
148           }
149           else if(payments[i, j] == 0 && 0 < x[sum(binom.c[ , j]) + 1,
     j] - 1 && x[sum(binom.c[ , j]) + 1, j] - 1 < p * b[sum(binom.c[ , j
    ]) + 1, j]){
150             payments[i, j] <- (x[sum(binom.c[ , j]) + 1, j] - 1) * B /
     b[sum(binom.c[ , j]) + 1, j]
151             break
152           }
153           else{
154             payments[i, j] <- payments[i, j]
155           }
156         }
157       }
158
159       vec.disc.v <- rep(0, npath)
160       for(j in 1:npath)
161       {
162         disc.v <- 0
163         int.r <- 0
164
165         for(i in 1:n)
166         {
167           int.r <- int.r + r[i, j] * dt
168           disc.v <- disc.v + exp(-int.r) * payments[i, j]
169         }
170         vec.disc.v[j] <- disc.v
171       }
172       V_t_sa <- mean(vec.disc.v)
173
174       reg.matrix[m, l] <- V_t_sa
175     }
176
177     reg.fit1[l] <- lm(reg.matrix[ , l]~1+c)$coef[1]
178     reg.fit2[l] <- lm(reg.matrix[,l]~1+c)$coef[2]
179
```

```
180         c.fit[l] <- (B-reg.fit1[l])/reg.fit2[l]
181       }
182       c.fit.matrix[,w] <- c.fit
183     }
184     return(V_t_sa)
185 }
186
187 # Pricing Example
188 price_coco_sa(T = 5, npath = 20, rho = - 0.2, kappa = 0.114, r.bar =
      0.069, r0 = 0.035, sigma_r = 0.07)
```