

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

1	Introduction and Motivation	1
1.1	Introduction	1
1.2	Literature Overview	1
1.3	Motivation	1
1.4	Methodology	1
2	Structure of CoCos	2
2.1	Description of CoCos	2
2.2	Payoff and Risk Profile	2
2.3	Conversion Trigger	2
2.3.1	Market Trigger	2
2.3.2	Accounting Trigger	2
2.3.3	Regulatory Trigger	2
2.3.4	Multivariate Trigger	2
2.4	Conversion Details	2
2.4.1	Conversion Fraction	2
2.4.2	Conversion Price and Ratio	2
3	Theory of Pricing	4
3.1	Credit Derivative Approach	4
3.1.1	Intensity-based Approach	4
3.1.2	Application to CoCos	5
3.1.3	Data Requirements and Calibration	5
3.1.4	Pricing Example	5
3.2	Equity Derivative Approach	5
3.2.1	Corporate Bonds	5
3.2.2	Binary Options	5
3.2.3	Down-And-In Forward	6

3.2.4	Data Requirements and Calibration	7
3.2.5	Pricing Example	7
4	Dynamics and Sensitivity Analysis	8
4.1	Credit Derivative Approach	8
4.2	Equity Derivative Approach	8
5	Empirical Analysis and Model Comparison	9
5.1	Data Description	9
5.1.1	Deutsche Bank	9
5.2	Model Parametrization	9
5.3	Model Comparison	9
5.3.1	Qualitative Analysis	9
5.3.2	Quantitative Analysis	9
6	Conclusion	10
A	Sample Title	11
B	Code - Credit Derivative Approach	12
C	Code - Equity Derivative Approach	14
D	Code - Structural Approach	17

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

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

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

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

$$R_{CoCo} = \frac{S^*}{C_p} \quad (3.4)$$

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

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

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

3.1.2 Application to CoCos

3.1.3 Data Requirements and Calibration

3.1.4 Pricing Example

3.2 Equity Derivative Approach

Sources: [?], [?]

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

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

3.2.1 Corporate Bonds

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

3.2.2 Binary Options

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

with

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

3.2.3 Down-And-In Forward

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

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

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

with

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

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

with

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

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

$$\min(S_t) > S^* : P_T = 0 \quad (3.16)$$

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

with

$$C_r = \frac{\alpha N}{C_p} \quad (3.18)$$

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.

```
1 # Price of Contingent Convertible Bond
2 price_coco_cd <- function(t, T, S_t, S_star, C_p, c_i, r, N, q, sigma){
3
4   spread_coco <- calc_spread_coco(t, T, S_t, S_star, C_p, r, q, sigma)
5   V_t_coco <- N * exp(-(r + spread_coco) * (T - t))
6
7   for (t in 1:T){
8     V_t_coco <- V_t_coco + c_i * exp(-(r + spread_coco) * t)
9   }
10  V_t_coco
11 }
12
13 # Calculation of Trigger Probability
14 calc_p_star <- function(t, T, S_t, S_star, r, q, sigma){
15   p_star <- pnorm((log(S_star / S_t) - calc_mu(r, q, sigma) * (T - t)) /
16     (sigma * sqrt(T - t))) + (S_star / S_t)^(2 * calc_mu(r, q, sigma) /
17     sigma^2) * pnorm((log(S_star / S_t) + calc_mu(r, q, sigma) * (T - t)
18     )) / (sigma * sqrt(T - t)))
19   p_star
20 }
21
22 # Calculation of Drift of Underlying
23 calc_mu <- function(r, q, sigma){
24   mu <- r - q - sigma^2 / 2
25   mu
26 }
27
28 # Spread of CoCo Bond
29 calc_spread_coco <- function(t, T, S_t, S_star, C_p, r, q, sigma){
30   spread_coco <- - log(1 - calc_p_star(t, T, S_t, S_star, r, q, sigma))
31     / (T - t) * (1 - S_star / C_p)
32   spread_coco
33 }
34
35 # Pricing Example
```

```
32 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.

```
1 # Price of Contingent Convertible Bond
2 price_coco_ed <- function(t, T, S_t, S_star, C_p, c_i, r, N, q, sigma,
  alpha){
3   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)
4
5   return(V_t_ed)
6 }
7
8 # Price of Corporate Bond
9 price_cb <- function(t, T, c_i, r, N){
10   V_t_cb <- N * exp(-r * (T - t))
11
12   for (t in 1:T){
13     V_t_cb <- V_t_cb + c_i * exp(-r * t)
14   }
15
16   return(V_t_cb)
17 }
18
19 # Price of Binary Option
20 price_dibi <- function(t, T, S_t, S_star, c_i, r, q, sigma, alpha){
21   V_t_dibi <- 0
22
23   i <- t
24   k <- T
25
26   for (i in 1:k) {
27     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)))
28   }
29 }
```

```

30 V_t_dibi <- alpha * V_t_dibi
31
32 return(V_t_dibi)
33 }
34
35 # Price of Down-And-In Forward
36 price_difwd <- function(t, T, S_t, S_star, C_p, r, N, q, sigma, alpha){
37   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)))
38
39   return(V_t_difwd)
40 }
41
42 # Calculation of Conversion Rate
43 calc_conversion_rate <- function(C_p, N, alpha){
44   C_r <- alpha * N / C_p
45
46   return(C_r)
47 }
48
49 # Calculation of additional Parameters
50 calc_x_1_i <- function(S_t, S_star, sigma, r, q, t_i){
51   x_1_i <- log(S_t / S_star) / (sigma * sqrt(t_i)) + calc_lambda(r, q,
    sigma) * sigma * sqrt(t_i)
52
53   return(x_1_i)
54 }
55
56 calc_y_1_i <- function(S_t, S_star, sigma, r, q, t_i){
57   y_1_i <- log(S_star / S_t) / (sigma * sqrt(t_i)) + calc_lambda(r, q,
    sigma) * sigma * sqrt(t_i)
58
59   return(y_1_i)
60 }
61
62 calc_lambda <- function(r, q, sigma){
63   lambda <- (r - q + sigma^2 / 2) / sigma^2
64
65   return(lambda)
66 }
67
68 calc_x_1 <- function(t, T, S_t, S_star, r, q, sigma){
69   x_1 <- log(S_t / S_star) / (sigma * sqrt(T - t)) + calc_lambda(r, q,
    sigma) * sigma * sqrt(T - t)
70
71   return(x_1)
72 }
73

```

```

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.

```
1 # Price of Contingent Convertible Bond
2 price_coco_sa <- function(T , npath , rho , kappa , r_bar, r0, sigma_r,
  mu_Y, sigma_Y, lambda, g, x_hat, b0, p, e_bar, sigma_x, x0_low, x0_
  high, x0_nint, B, c_low, c_high, c_nint){
3   n <- T * 250
4   dt <- T / n
5
6   result <- sim_corrProcess(T, npath, rho, n, dt)
7   dW_1 <- result$dW_1
8   dW_2corr <- result$dW_2corr
9
10  r <- sim_interestrates(kappa, r_bar, r0, sigma_r, dW_2corr, n, npath,
  dt)
11
12  V_t_sa <- get_price(npath, n, dt, dW_1, dW_2corr, r, mu_Y, sigma_Y,
  lambda, g, x_hat, b0, p, e_bar, sigma_x, x0_low, x0_high, x0_nint, B
  , c_low, c_high, c_nint) * 100
13  return(V_t_sa)
14 }
15
16 sim_corrProcess <- function(T, npath, rho, n, dt){
17   vect <- c(1, rho, rho, 1)
18   RHO <- matrix(vect, nrow = 2)
19   chol_RHO <- t(chol(RHO))
20
21   # Create two Brownian Motions
22   dW_1 <- matrix(1, n, npath)
23   dW_2 <- matrix(1, n, npath)
24
25   for(j in 1:npath)
26   {
27     dW_1[, j] <- rnorm(n) * sqrt(dt)
28     dW_2[, j] <- rnorm(n) * sqrt(dt)
29   }
30 }
```



```

31 # Create Correlated Process based on Brownian Motions using Cholesky-
    Decomposition
32 dW_2corr <- matrix(1, n, npath)
33 for(j in 1:npath)
34 {
35   for(i in 1:n)
36   {
37     dW_2corr[i, j] <- dW_1[i, j] * chol_RHO[2, 1] + dW_2[i, j] * chol_
        RHO[2, 2]
38   }
39 }
40
41 return(list("dW_1" = dW_1, "dW_2corr" = dW_2corr))
42 }
43
44 # Create Interest Rate Process
45 sim_interestrates <- function(kappa, r_bar, r0, sigma_r, dW_2corr, n,
    npath, dt){
46   r <- matrix(r0, n + 1, npath)
47
48   for(j in 1:npath)
49   {
50     for(i in 1:n)
51     {
52       r[i + 1, j] <- r[i, j] + kappa * (r_bar - r[i, j]) * dt + sigma_r
        * sqrt(r[i, j]) * dW_2corr[i, j]
53     }
54   }
55
56   return(r)
57 }
58
59 get_price <- function(npath, n, dt, dW_1, dW_2corr, r, mu_Y, sigma_Y,
    lambda, g, x_hat, b0, p, e_bar, sigma_x, x0_low, x0_high, x0_nint, B
    , c_low, c_high, c_nint){
60
61   c_fit_matrix <- matrix(0, x0_nint, length(lambda))
62
63   for(w in 1:length(lambda))
64   {
65     # Create parametres for jump process
66     phi <- matrix(rbinom( n%*%npath, 1, dt * lambda[w]), n, npath)
67     ln_Y <- matrix(rnorm(n%*%npath, mu_Y, sigma_Y), n, npath)
68
69     b <- matrix(b0, n + 1, npath)
70     x_bar0 <- 1 + e_bar + p * b0
71     x_bar <- matrix(x_bar0, n + 1, npath)
72
73     h <- matrix(1, n, npath)
74
75     k <- exp(mu_Y + 0.5 * sigma_Y^2) - 1
76
77     c <- seq(c_low, c_high, length = c_nint)
78     x0 <- seq(x0_low, x0_high, length = x0_nint)

```

```

79   for(l in 1:x0_nint) # Wieso?
80   {
81     for(m in 1:c_nint) # Wieso?
82     {
83       x <- matrix(x0[l],n+1,npath)
84       ln_x0 <- matrix(log(x0[l]),n+1,npath)
85       ln_x <- ln_x0
86       binom_c <- matrix(1,n+1,npath)
87
88       for(j in 1:npath)
89       {
90         for(i in 1:n)
91         {
92           d_1 <- (ln_x[i, j] + mu_Y) / sigma_Y
93           d_2 <- d_1 + sigma_Y
94
95           h[i, j] <- lambda[w] * (pnorm(- d_1) - exp(ln_x[i, j]) *
96             exp(mu_Y + 0.5 * sigma_Y^2) * pnorm(-d_2))
97
98           b[i + 1, j] <- b[i, j] * exp(- g[w] * (exp(ln_x[i, j]) - x_
99             hat) * dt)
100
101           ln_x[i + 1, j] <- ln_x[i, j] + ( (r[i, j] - lambda[w] * k) -
102             (r[i, j] + h[i, j] + c[m] * b[i, j]) / exp(ln_x[i, j]) - g[w] * (
103             exp(ln_x[i, j]) - x_hat) - 0.5 * sigma_x^2) * dt + sigma_x * sqrt(dt
104             ) * dW_1[i, j] + ln_Y[i,j] * phi[i, j]
105
106           x[i + 1, j] <- exp(ln_x[i + 1, j])
107
108           x_bar[i + 1, j] <- 1 + e_bar + p * b[i + 1, j]
109
110           if(x[i + 1, j] >= x_bar[i + 1, j] && binom_c[i, j] > 0.5)
111           {
112             binom_c[i + 1, j] <- 1
113           }else
114           {
115             binom_c[i + 1, j] <- 0
116           }
117         }
118       }
119
120       payments <- matrix(c(rep(c[m] * dt, n - 1), B), n, npath) *
121       binom_c[1:n, ]
122
123       for(j in 1:npath){
124         for(i in 2:n){
125           if(payments[i, j] == 0 && p * b[sum(binom_c[, j]) + 1, j]
126             <= x[sum(binom_c[, j]) + 1, j] - 1 ){
127             payments[i, j] <- p * B
128             break
129           }
130           else if(payments[i, j] == 0 && 0 < x[sum(binom_c[, j]) + 1,
131             j] - 1 && x[sum(binom_c[, j]) + 1, j] - 1 < p * b[sum(binom_c[, j]

```

```

125     )) + 1, j)) {
126         payments[i, j] <- (x[sum(binom_c[, j]) + 1, j] - 1) * B /
127         b[sum(binom_c[, j]) + 1, j]
128         break
129     }
130     else {
131         payments[i, j] <- payments[i, j]
132     }
133 }
134 vec_disc_v <- rep(0, npath)
135 for(j in 1:npath)
136 {
137     disc_v <- 0
138     int_r <- 0
139     for(i in 1:n)
140     {
141         int_r <- int_r + r[i, j] * dt
142         disc_v <- disc_v + exp(- int_r) * payments[i, j]
143     }
144     vec_disc_v[j] <- disc_v
145 }
146
147 V_t_sa <- mean(vec_disc_v)
148
149 return(V_t_sa)
150 }
151 }
152 }
153 }
154
155 # Pricing Example
156 price_coco_sa(T = 5, npath = 2, rho = - 0.2, kappa = 0.114, r_bar =
    0.069, r0 = 0.035, sigma_r = 0.07, mu_Y = -0.01, sigma_Y = 0.02,
    lambda = c(1), g = c(0.5), x_hat = 1.1, b0 = 0.04, p = 1, e_bar =
    0.02, sigma_x = 0.02, x0_low = 1.15, x0_high = 1.15, x0_nint = 10, B
    = 1, c_low = 0.05, c_high = 0.05, c_nint = 10)

```