# 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

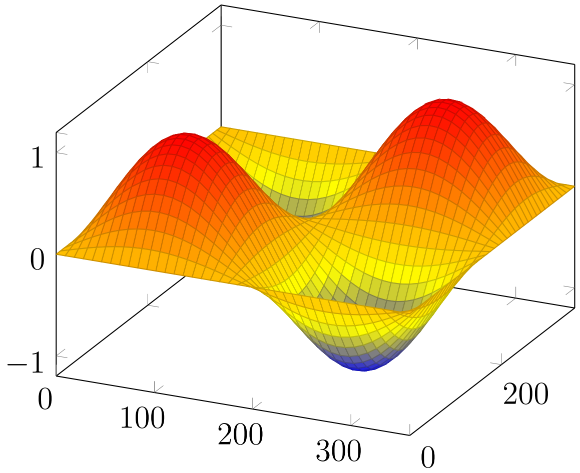
### 2.3.4 Multivariate Trigger

## 2.4 Conversion Details

### 2.4.1 Conversion Fraction

- conversion fraction $\alpha$
- 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

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

### 3.2.2 Binary Options

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

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

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

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

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

with

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

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

with

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

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

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

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

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

# Sample Title

The next code will be directly imported from a file

```
library(magrittr)

### Plain-Vanilla Black-Scholes

bs <- function(spot, strike, r_d, r_f, sigma, maturity, callput = 1) {
  forward <- spot * exp((r_d - r_f) * maturity)
  d_plus <- (log(forward / strike) + sigma ** 2 / 2 * maturity) / (sigma * sqrt(maturity))
  d_minus <- (log(forward / strike) - sigma ** 2 / 2 * maturity) / (sigma * sqrt(maturity))
  exp(-r_d * maturity) * callput * (forward * pnorm(callput * d_plus) - strike * pnorm(callput * d_minus))
}

plain_vanilla_call <- function(spot, strike, r_d, r_f, sigma, maturity) {
  bs(spot, strike, r_d, r_f, sigma, maturity, 1)
}

plain_vanilla_put <- function(spot, strike, r_d, r_f, sigma, maturity) {
  bs(spot, strike, r_d, r_f, sigma, maturity, -1)
}

### Double Knock-Out Options
# Pricing Options with curved boundaries, Naoto Kunitomo, Masayuki Ikeda (1992)
# case for delta_1=delta_2=0

dko <- function(spot, strike, upper, lower, r_d, r_f, sigma, t, callput = 1) {
  stopifnot(callput %in% c(-1,1))
  if (or(spot < lower, spot > upper)) { return(0)}
  r_prime <- (r_d - r_f + sigma ** 2 / 2) * t
  sigma_scaled <- sigma * sqrt(t)
  c1 <- 2 * (r_d - r_f) / (sigma ** 2) + 1
  c3 <- c1
  c2 <- c1 + 1
```

```
  -5:5 %>% sapply(function(n){
    if (callput == -1) {
      d1n <- (log(spot * upper ** (2 * n)  / (lower ** (2 * n + 1)))          + r_prime) / sigma_scaled
      d2n <- (log(spot * upper ** (2 * n)  / (strike * lower ** (2 * n)))          + r_prime) / sigma_scaled
      d3n <- (log(lower ** (2 * n + 2)      / (lower * spot * upper ** (2 * n)))  + r_prime) / sigma_scaled
      d4n <- (log(lower ** (2 * n + 2)      / (strike * spot * upper ** (2 * n))) + r_prime) / sigma_scaled
    }
    if (callput == 1) {
      d1n <- (log(spot * upper ** (2 * n)  / (strike * lower ** (2 * n)))           + r_prime) / sigma_scaled
      d2n <- (log(spot * upper ** (2 * n - 1) / (lower ** (2 * n)))           + r_prime) / sigma_scaled
      d3n <- (log(lower ** (2 * n + 2)      / (strike * spot * upper ** (2 * n)))  + r_prime) / sigma_scaled
      d4n <- (log(lower ** (2 * n + 2)      / (spot * upper ** (2 * n + 1)))        + r_prime) / sigma_scaled
    }
    c2n <- 2 * n / sigma ** 2

    sum1 <- (upper ** n / lower ** n) ** c1 * (pnorm(d1n) - pnorm(d2n)) -
      ((lower ** (n + 1)) / (upper ** n * spot)) ** c3 * (pnorm(d3n) - pnorm(d4n))
    sum2 <- (upper ** n / lower ** n) ** (c1 - 2) * (pnorm(d1n - sigma_scaled) - pnorm(d2n - sigma_scaled)) -
      ((lower ** (n + 1)) / (upper ** n * spot)) ** (c3 - 2) * (pnorm(d3n - sigma_scaled) - pnorm(d4n - sigma_scaled))

    c(sum1, sum2)
  }) %>% rowSums %>%  (function(sums) {
    (callput * c(spot * exp(-r_f * t), -strike * exp(-r_d * t))) %*% sums
  })
}

dko_call <- function(spot, strike, upper, lower, r_d, r_f, sigma, t) {
  dko(spot, strike, upper, lower, r_d, r_f, sigma, t, 1)
}

dko_put <- function(spot, strike, upper, lower, r_d, r_f, sigma, t) {
  dko(spot, strike, upper, lower, r_d, r_f, sigma, t, -1)
}

####

#dko(1.1, 1.10522, 1.155, 1.045, 0.5E-2, -0.2E-2, 11.08E-2, 0.5, -1)

package <- function(spot = 1.1, strike = 1.12, upper=1.21, lower=1.02, leverage=1, maturity=0.5) {
  r_d <- 0.5E-2
```

```
  r_f <- -0.2E-2
  sigma <- 11.08E-2
  result <- leverage * dko_call(spot, strike, upper, lower, r_d, r_f, sigma, maturity) +
            leverage * dko_put(spot, strike, upper, lower, r_d, r_f, sigma, maturity) +
              plain_vanilla_call(spot, strike, r_d, r_f, sigma, maturity) -
              plain_vanilla_put(spot, strike, r_d, r_f, sigma, maturity)
  result
}

dko_package <- function(spot=1.1, sigma=11.08E-2, strike=1.13351, upper=1.21, lower=0.99) {
  r_d <- 0.5E-2
  r_f <- -0.2E-2
  maturity <- 0.5
  result <- 100 * dko_call(spot, strike, upper, lower, r_d, r_f, sigma, maturity) +
    100 * dko_put(spot, strike, upper, lower, r_d, r_f, sigma, maturity)
  result
}

#dko_package2 <- function(spot=1.1, sigma=11.08E-2, strike=1.13351, dist=0.05) {
#  r_d <- 0.5E-2
#  r_f <- -0.2E-2
#  maturity <- 0.5
#  forward <- 1.10395
#  worst_case <- strike
#  upper <- worst_case + dist
#  lower <- worst_case - dist
#  result <- 100 * dko_call(spot, strike, upper, lower, r_d, r_f, sigma, maturity) +
#    100 * dko_put(spot, strike, upper, lower, r_d, r_f, sigma, maturity) +
#    plain_vanilla_call(1.1, strike, r_d, r_f, sigma, maturity) -
#    plain_vanilla_put(1.1, strike, r_d, r_f, sigma, maturity)
#  result
#}

########### Zero Cost
#dko_package2(strike = 1.12, dist = 0.04531) #0 EUR
#dko_package2(strike = 1.12, dist = 0.04535) #2k EUR
###########

spot <- seq(1,1.25,0.005)
vol <- 11.02E-2
spot %>% sapply(function(x)dko_package(x,vol)) %>% plot(spot, y = ., type = "l", main = "Value")
spot %>% sapply(function(x)dko_package(x,vol)) %>% diff %>% plot(spot[-1], y = ., type = "l", main = "Delta")

vol <- seq(0.5E-2,30E-2,0.005)
```

```r
vol %>% sapply(function(x)dko_package(sigma = x)) %>% plot(vol, y = ., type = "l", main = "Value")
vol %>% sapply(function(x)dko_package(sigma = x)) %>% diff %>% plot(vol[-1], y = ., type = "l", main = "Vega")


######

spot <- seq(0.9,1.35,0.005)
spot %>% sapply(function(x)package(spot = x, upper = 1.2190, lower = 1.0230)) %>% data.frame(.,spot)
  plot(spot, y = ., type = "l", main = "Value")

writeOut <- function(name, fn) {
  spot %>% sapply(fn) %>% multiply_by(1E7) %>% data.frame(spot,value=.) %>% write.table(file = name, na = "nan", row.names = F, quote = F)
}

writeOut("unlev", function(x)package(spot = x, upper = 1.2190, lower = 1.0230))
writeOut("unlev_margin", function(x)package(spot = x, upper = 1.2190, lower = 1.0243))
writeOut("lev", function(x)package(spot = x, upper = 1.1792, lower = 1.0608, leverage = 10))
writeOut("lev_margin", function(x)package(spot = x, upper = 1.1789, lower = 1.0611, leverage = 10))

## ----- OpenGL 3D
library(rgl)
spot <- seq(0.95,1.25,0.005)
vol <- seq(0.5E-2,15E-2,0.005)
l <- list(spot, vol)
comb <- do.call(expand.grid, l)
colnames(comb) <- c("Spot", "Vol")
head(comb)
value <- mapply(function(Spot, Vol) dko_package(Spot, Vol, strike=1.12, upper=1.2190, lower=1.0230), comb$Spot, comb$Vol)

result <- cbind(comb, value)
result %>% write.table(file = "3dplot", na = "nan", row.names = F, quote = F)
#wireframe(value ~ Spot * Vol, result)
persp3d(spot, vol, result$value, col = "skyblue",xlab = "Spot", ylab="Volatility", zlab="Value")
## ------

time <- seq(1/12,1,1/12)

c <- do.call(expand.grid, list(spot, time))
colnames(c) <- c("Spot", "Time")

value_delta <- mapply(function(Spot, Time) dko_package(Spot, Time, strike=1.12, upper = 1.2190, lower = 1.0230), c$Spot, c$Time)
result_delta <- cbind(c, value_delta)
```

```
idx <- result_delta %>% group_by(Time) %>% attr(.,"indices")

deltas <- idx %>% lapply(function(indices){
  result_delta[indices,] %>% (function(x) diff(x$value_delta))
})

c_merged <- do.call(expand.grid, list(spot[-1], time))
colnames(c_merged) <- c("Spot", "Time")
final_result_delta <- cbind(c_merged, delta=unlist(deltas))
final_result_delta %>% write.table(file = "delta_surface", na = "nan",
    row.names = F, quote = F)
persp3d(spot[-1], time, final_result_delta$delta, col = "skyblue",xlab =
     "Spot", ylab="Volatility", zlab="Value")
```