The Costs of Counterparty Risk in Long Term Contracts Code Guide - Section 3

Michael Duarte Gonçalves

July 4, 2025

Overview

This file provides a detailed overview of the code structure for the numerical methods exercise. It is intended to guide future contributors through the logic, organization, and rationale of each section and function.

Contents

1 Modeling Markets Without Counterparty Risk (CPR)

1.1 Theoretical Methodology

Modelling Prices We assume that $p \in (0, 1)$ and $f \in (0, 1)$. To scale them up, they are both multiplied by a parameter x, which we set at $x = 60$ but could consider alternative values. Spot prices and contract prices are thus px and fx. We assume that spot prices p follow a beta distribution[1] with $\alpha = 4$ and $\beta = 2$. Since this function takes values in $[0, 1]$, spot prices px take value in $[0, 60]$:

$$\mathbb{E}(p) = \frac{\alpha}{\alpha + \beta} = \frac{2}{3}, \quad \text{Var}(p) = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)} = \frac{2}{63}.$$

Spot market profits Sellers get the following utilities when trading in the spot market:

$$\Pi_S^0(c) = q_i x \mathbb{E}(p) - C(k_i) - r_i, \tag{1}$$

where:

q_i is the total production of plant i, in MWh;

x represents the scaling factor;

$\mathbb{E}(p)$ displays the expected price;

$C(k_i)$ shows total costs of plant i;

$r_i = r_0(xq_i)^2 Var(p)$.

(i) We compute spot profits for all plants.

(ii) We want to make sure that some plants find it optimal to trade in the spot market, so we choose r_0 sufficiently small so that, for at least some plants, $\Pi_S^0(c) > 0$.

(iii) To do this, we choose a sufficiently low value for r_0 such that a subset of the plants have positive spot market profits. The sum of their production is q_0.

Contracts with no CPR With contracts, sellers make profits:

$$\Pi_S(f) = q_i x f - C(k_i). \tag{2}$$

Contract prices need to satisfy two constraints:

1. Break-even constraint: contract profits (2) have to be non-zero, i.e., xf has to be above average cost. Hence, for each plant, compute

$$xf^c \geq \frac{C(k_i)}{q_i} \tag{3}$$

which is equal to the plant's average cost.

2. Spot market constraint: xf has to such that the contract is more profitable than trading in the spot market, i.e., contract profits (2) have to be greater or equal than (1). Hence, for each plant, we compute:

$$xf^{spot} \geq xE(p) - r_0 Var(p) q_i x^2 \tag{4}$$

[1] For shape parameters $\alpha = 4$ and $\beta = 2$, the hazard rate is strictly increasing.

(a) For each plant, we find the max between xf^c and xf^{spot}. *Note:* that all those plants with positive spot market profits have $xf^{spot} > xf^c$;

(b) We rank plants in increasing order according to the maximum between xf^c, xf^{spot} and plot the cumulative curve using the plant's production (contract supply curve). For instance, suppose that we have two plants: plant A has $xf^c = 10, xf^{spot} = 20$ and production 100 and plant B has $xf^c = 30, xf^{spot} = 10$ and production 50. Then, the curve is $p = 20$ up to quantity 100 and $p = 30$ from 100 to 150.

Important: Recall that r_0 is defined in the `Section1_data_creation.tex` file and is fixed to $r_0 = 1.334653 \times 10^{-7}$. Below, we use a function to automatically compute the r_0, based on how many plants we want to make positive profits in the spot market (1%, 2%, *etc*). But this function is never used in the code (it is commented out, as you will see).

We therefore use the same r_0 for all simulations, of course. The function is shown for completeness and if you need to use it once.

1.2 Numerical Methods

1.2.1 Spot Market Profits Without CPR

Purpose: To compute the profits that each project would earn in the spot market, accounting for risk, and to calibrate the risk parameter r_0 so that a chosen fraction of projects (e.g., 1%) achieve positive profits.

How:

- Define a function `check_profit_proportion_sp_no_cpr` that, for a given r_0, computes the share of projects with positive profits.
- Use a root-finding function `find_optimal_r0` to determine the minimum r_0 such that at least the target proportion (e.g., 1%) of projects are profitable.
- Update the dataset with the calculated r_0, risk adjustment r_i, and spot market profits for each project.

Important:

- You can skip the Listing below, since it is creating a function we do not need, as we already have our r_0 computed and fixed. But for completeness, below are some explanations about all this.

Listing 1: Spot market profit calculation and r_0 calibration

```
# Section 3: Modelling Markets w/o CPR --------------------- #

# INSTRUCTIONS! ------------------------------------------- #
# We will now modeling markets without CPR
  # Spot market profits

    # Sellers get the following utilities:

      #  i.  \Pi_S^0(c) = q_i \times x \times E(p) - C(k_i) - r_i,
      #  ii. r_i = r_0 Var(pxq_i) = r_0(xq_i)^2 Var(p)

      #  iii. Be careful! r_0 should be sufficiently low, in order to
      #  make sure that some plants find it optimal to trade in spot m.
      #  for at least some plants, we should have \Pi_S^0(c) > 0.


#  ---------------------------------------------------------- #
#  ---------------------------------------------------------- #

# Functions

# Define some functions. Here, we want to make sure that at least some
   plants make profits in the spot market. For that, we will use some
# numerical methods to find this minimal threshold

# N.B.: acronyms "sp" means "spot market" and
# "no_cpr" below means "no counterparty risk"

# Define function to check profit percentage for a given r_0
```

```
check_profit_proportion_sp_no_cpr <- function(r_0, df) {
  df <- df |>
    mutate(r = r_0 * (x * q_i_mwh)^2 * var_p,  # Calculate r_i
           profits_sp_no_cpr = q_i_mwh * x * expected_p - total_cost -
              r)

  # Calculate the percentage of projects with positive profits
  mean(df$profits_sp_no_cpr > 0)
}

#  ---------------------------------------------------------- #
#  ---------------------------------------------------------- #

# Function: Find the smallest r_0 that ensures at least 1% of projects
   are profitable
find_optimal_r0 <- function(df, lower = 1e-12, upper = 1e-3, target =
   0.01, tol = 1e-12) {
  # Define function to find zero crossing
  f <- function(r_0) check_profit_proportion_sp_no_cpr(r_0, df) -
     target

  # Use uniroot instead of optimize
  result <- tryCatch(
    uniroot(f, interval = c(lower, upper), tol = tol)$root,
    error = function(e) return(NA)  # Return NA if no solution is found
  )

  return(result)
}

#  ---------------------------------------------------------- #
#  ---------------------------------------------------------- #

 Spot Market Profits

# Based on the previous created function, one could choose
# approximately the percentage of the sample that we want to have
# positive profits, by changing above the target value that we want.

# Example: now, the target is defined to 0.01, meaning that we want 1%
# of the sample to have positive profits.

# This is the aim of the following command. Compute the optimal r_0
   given
# a target:

# NOTE: IF YOU WANT TO USE THIS r_0, please delete the one defined
   above
# in "Key Parameters" subsection of Section 1.

# r_0 <- find_optimal_r0(wind_solar_proj_2022)

wind_solar_proj_2022 <- wind_solar_proj_2022 |>
  mutate(r_0 = r_0,
         r = r_0 * (x*q_i_mwh)^2 * var_p,
         profits_sp_no_cpr = q_i_mwh * x * expected_p - total_cost - r
  )
```

```
# Check final profit percentage
profit_percentage <- mean(wind_solar_proj_2022$profits_sp_no_cpr > 0)*
    100
cat("Final percentage of profitable projects:", profit_percentage, "%\n
    ")
```

Outcome: The dataset now contains, for each project, the spot market profit after risk adjustment. The final percentage of profitable projects is reported to verify the calibration.

1.2.2 Understanding `find_optimal_r0`, Interval, and Tolerance

The function `find_optimal_r0` is designed to numerically determine the smallest value of the risk parameter r_0 such that at least a target proportion (e.g., 1%) of projects in the dataset are profitable in the spot market. This is achieved using the root-finding algorithm `uniroot` in R.

How `find_optimal_r0` Works?

It defines a function $f(r_0)$ that computes the difference between the proportion of profitable projects (given r_0) and the target proportion.

It then uses `uniroot` to search for the value of r_0 where $f(r_0) = 0$, i.e., where the actual proportion matches the target.

If no solution is found within the specified interval, the function safely returns `NA`.

The Role of `interval = c(lower, upper)`

The `interval` argument sets the search range for r_0.

The Role of `tol`

The `tol` argument sets the numerical precision for the solution.

`tol = 1e-12` means the algorithm will stop when the estimated root is within 10^{-12} of the actual root.

Lower values of `tol` produce more precise results but may require more computation time.

What is `uniroot`?

`uniroot` is an R function that finds a root (zero) of a continuous function within a specified interval.

In this context, it finds the value of r_0 for which the proportion of profitable projects equals the target.

It requires that the function changes sign over the interval (i.e., $f(\text{lower})$ and $f(\text{upper})$ have opposite signs).

1.3 Contract Supply Curve Construction Without CPR

This section details the construction of the contract supply curve in the absence of counterparty risk (CPR). The process involves defining project-specific price constraints, aggregating supply, and identifying market equilibrium.

1.4 Project-Level Contract Price Constraints

For each project, we compute two critical price thresholds that determine their willingness to enter contracts:

$$f^c = \frac{C(k_i)}{q_i x} \qquad \text{(Break-even constraint)}$$

$$f^{\text{spot}} = E(p) - r_0 \cdot \text{Var}(p) \cdot q_i \cdot x \qquad \text{(Spot market constraint)}$$

The binding constraint for each project is the maximum of these two prices:

$$f_{\max} = \max(f^c, f^{\text{spot}})$$

Listing 2: Project-level price constraints

```
wind_solar_proj_2022_no_cpr <- wind_solar_proj_2022 |>
  mutate(
    f_c     = total_cost / (q_i_mwh * x),  # Break-even const.
    f_spot = (x * expected_p - r_0 * var_p * q_i_mwh * x^2) / x,   #
       Spot const.
    f_max = pmax(f_c, f_spot),
    xf_c = x * f_c,
    xf_spot = f_spot * x,
    xf_max  = pmax(xf_c, xf_spot) # Take the max of the two
  )
```

1.5 Identifying Binding Constraints

Listing 3: Constraint binding identification

```
# Identify the chosen xf for positive profits
wind_solar_proj_2022_no_cpr <- wind_solar_proj_2022_no_cpr |>
  mutate(
    chosen_xf = case_when(
      profits_sp_no_cpr > 0 & xf_max == xf_c ~ "xf_c",
      profits_sp_no_cpr > 0 & xf_max == xf_spot ~ "xf_spot",
      TRUE ~ "xf_c"
    )
  ) |>
  mutate(x = x,
         expected_p = expected_p,
         var_p = var_p) |>
  select(projectname, capacity, x, expected_p, var_p, everything())
```

Key Insight: Projects with $\Pi_S^0 > 0$ always have $f^{\text{spot}} > f^c$, meaning their contract participation is constrained by spot market opportunity costs rather than break-even requirements.

Construct Supply Curve Projects are sorted by f_{max} and aggregated:

1. Group projects by f_{max}
2. Sum capacity and production within each price tier
3. Compute cumulative supply
4. Calculate $G_{\text{expected\_p\_x}}$: Capacity with $f_{\text{max}} \leq x \cdot E(p)$

Listing 4: Supply curve aggregation

```
contract_supply_nocpr <- wind_solar_proj_2022_no_cpr |>
  group_by(xf_max) |>
  summarise(
    total_capacity = sum(capacity),       # Sum capacity for this
        contract price
    total_production = sum(q_i_mwh)      # Sum production for this
        contract price
  ) |>
  arrange(xf_max) |>                      # Ensure ordering by price
  mutate(
    cumulative_capacity = cumsum(total_capacity),      # Cumulative sum
        of capacity
    cumulative_production = cumsum(total_production), # Cumulative sum
        of total production
    q_0 = q_0,
    G_expected_p_x = cumsum(if_else(xf_max <= expected_p * x,
                                total_capacity, 0))
  ) |>
  mutate(
    G_expected_p_x = last(G_expected_p_x, order_by = xf_max)
  ) |>
  ungroup()
```

1.6 Visualization of Supply Curve

Two complementary visualizations are created:

1. Cumulative production vs. contract price
2. Cumulative capacity vs. contract price

Listing 5: Supply curve visualization

```
# Plot cumulative production vs. xf_contract
cumul_prod_xfmax <- ggplot() +
  # Contract Supply Curve (Step Function)
  geom_step(data = contract_supply_nocpr, aes(x = cumulative_production
      , y = xf_max),
          color = theme_palette_avg_cost_graphs, size = 1) +
  # Labels and theme
  labs(x = "Cumulative Production (MWh)",
       y = expression("Contract Price (EUR/MWh)")) +
  theme_minimal(base_size = base_s) +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold")
  )
```

```
# Plot cumulative capacity vs. xf_contract
cumul_cap_xfmax <- ggplot() +
  # Contract Supply Curve (Step Function)
  geom_step(data = contract_supply_nocpr, aes(x = cumulative_capacity,
     y = xf_max),
            color = theme_palette_avg_cost_graphs, size = 1) +
  # Labels and theme
  labs(x = "Cumulative Capacity (MW)",
       y = "") +
  theme_minimal(base_size = base_s) +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold")
  )

cumul_graphs <- cumul_prod_xfmax + cumul_cap_xfmax

cumul_graphs

cumul_supply_path <- file.path(no_cpr_base_fig_dir, "01_supply_no_cpr.
   pdf")

ggsave(cumul_supply_path, plot = cumul_graphs, width = 16, height = 9,
   dpi = 300)
```

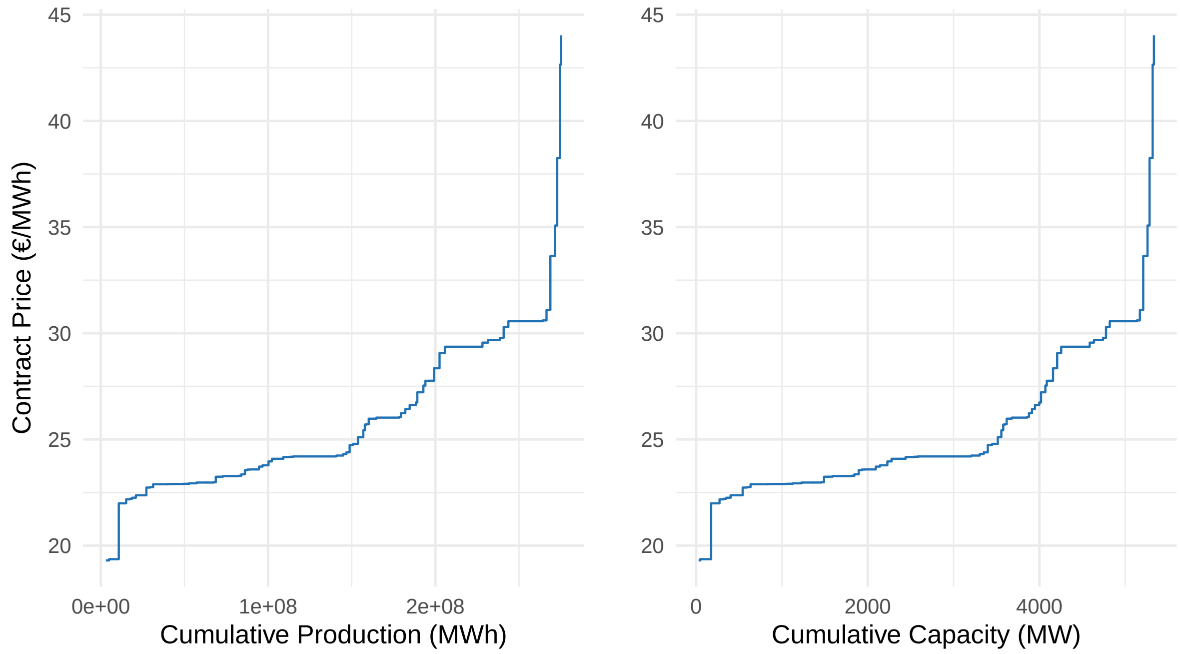

Figure 1: Supply Curves w/o CPR

1.7 Supply-Demand Equilibrium

Contract demand is fixed at $\theta = 3500$ MW

Equilibrium Identification The equilibrium price f^* is the minimum price where cumulative capacity $\geq \theta$:

$$f^* = \min\{f_{\max} : G_k(f_{\max}) \geq \theta\}$$

Visualization The equilibrium is visualized by overlaying demand on the supply curve:

Listing 6: Equilibrium visualization

```
# Equilibrium Prices/Quantities

# Create demand segment for plotting
demand_segments <- tibble(
  x_start = theta_no_cpr,
  x_end   = theta_no_cpr,
  y_start = min(contract_supply_nocpr$xf_max),
  y_end   = max(contract_supply_nocpr$xf_max)
)

# Generate the plot
supply_demand_plot <- ggplot() +
  geom_step(data = contract_supply_nocpr, aes(x = cumulative_capacity,
      y = xf_max),
            color = theme_palette_avg_cost_graphs, size = 1) +
  geom_segment(data = demand_segments, aes(x = x_start, xend = x_end,
                                            y = y_start, yend = y_end),
               color = "black", size = 1, linetype = "solid") +
  labs(x = "Cumulative Capacity (MW)",
       y = expression("Contract Price (EUR/MWh)")) +
  theme_minimal(base_size = base_s) +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold")
  )

supply_demand_plot

# Save the plot
plot_path <- file.path(no_cpr_base_fig_dir, "02_supply_demand_no_cpr.
    pdf")
ggsave(plot_path, plot = supply_demand_plot, width = 16, height = 9,
    dpi = 300)


equilibrium_nocpr <- contract_supply_nocpr |>
  filter(cumulative_capacity >= theta_no_cpr) |>
  slice(1) |>
  mutate(equilibrium_quantity = theta_no_cpr) |>  # Store theta as
      equilibrium quantity
  select(xf_max, equilibrium_quantity) |>
  rename(equilibrium_price = xf_max)
```

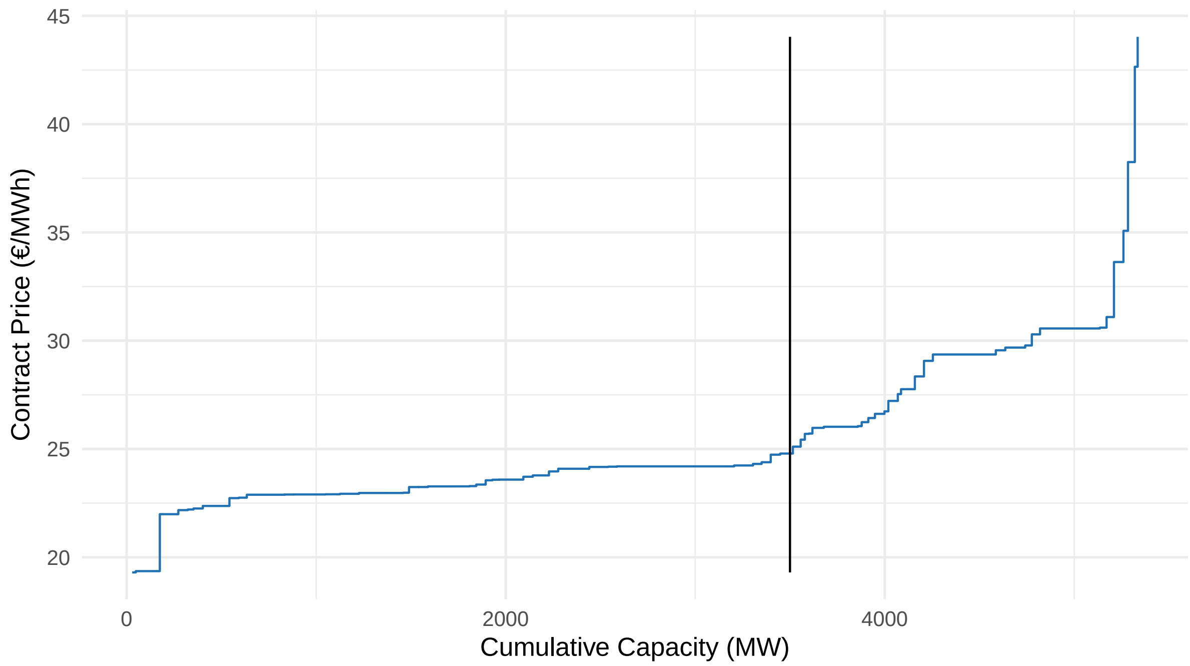

Figure 2: Equilibrium where supply meets contract demand $\theta = 3,500$

1.8 Data Export

All key datasets are exported for reproducibility:

- Project-level data with constraints
- Aggregated supply curve
- Equilibrium results

Listing 7: Data export

```
# Save in Excel format

wind_solar_proj_2022_no_cpr <- wind_solar_proj_2022_no_cpr |>
  mutate(theta = theta_no_cpr)

# Define the path
excel_path_s_w_proj <- file.path(no_cpr_tab_dir, "wind_solar_proj_2022"
    , "01_wind_solar_proj2022_nocpr.xlsx")

# Create workbook
wb_nocpr <- createWorkbook()

# Add first sheet: wind_solar_proj_2022_no_cpr
addWorksheet(wb_nocpr, "Wind_Solar_Projects")
freezePane(wb_nocpr, "Wind_Solar_Projects", firstActiveRow = 2,
    firstActiveCol = 2)
writeData(wb_nocpr, sheet = "Wind_Solar_Projects", x = wind_solar_proj_
    2022_no_cpr)

# Add second sheet: equilibrium_prices_no_ratio
addWorksheet(wb_nocpr, "Equilibrium_P_Q")
freezePane(wb_nocpr, "Equilibrium_P_Q", firstActiveRow = 2,
    firstActiveCol = 2)
writeData(wb_nocpr, sheet = "Equilibrium_P_Q", x = equilibrium_nocpr)

# Save workbook
saveWorkbook(wb_nocpr, file = excel_path_s_w_proj, overwrite = TRUE)
```

1.9 Key Variables

- The supply curve is stepwise due to discrete projects
- Projects with spot market profits have higher reservation prices
- Equilibrium price depends on both technology costs and spot market dynamics
- q_0 represents the opportunity cost threshold for contract participation

Table 1: Description of Variables from sheet `Wind_Solar_Projects`

Variable	**Description**
`projectname`	Name of the wind or solar project
`capacity`	Plant capacity (MW)
`x`	Price scaling factor
`expected_p`	Expected value of p ($\mathbb{E}(p) = \frac{2}{3}$)
`var_p`	Variance of p ($\mathbb{V}(p) = \frac{2}{63}$)
`avgcapacityfactor`	Average capacity factor (fraction of time producing)
`type`	Technology type (`Solar` or `Wind`)
`hours`	Annual operational hours (= `avgcapacityfactor` × 8760)
`power_kw`	Plant capacity in kilowatts (kW)
`q_i_kwh`	Lifetime production in kilowatt-hours (kWh), q_i
`q_i_mwh`	Lifetime production in megawatt-hours (MWh), q_i
`v_q_i_mwh`	Economic value of production (€/MWh)
`c_inv`	Investment cost per kW (€)
`c_om`	Operation & maintenance cost per kW (€)
`total_cost`	Total cost over plant lifetime $C(k_i)$ (€)
`avg_cost_euro_kwh`	Average cost per kWh (€)
`avg_cost_euro_mwh`	Average cost per MWh (€)
`r_0`	Calibrated risk aversion parameter, fixed ($r_0 \approx 1.33 \cdot 10^{-7}$)
`r`	Project-specific risk cost (€), r_i
`profits_sp_no_cpr`	Spot market profit without CPR (€), Π_S^0
`f_c`	Break-even contract price (€/MWh), f^c
`f_spot`	Minimum contract price for spot market participation (€/MWh), f^{spot}
`f_max`	Binding contract price constraint (€/MWh), f_{max}
`xf_c`	xf_c (scaled break-even contract price)
`xf_spot`	xf_{spot} (scaled spot market price)
`xf_max`	Maximum of `xf_c` and `xf_spot`
`chosen_xf`	Indicates which constraint is binding (`xf_c` or `xf_spot`)
`theta`	Contract demand parameter (here: $\theta = 3500$)