The Costs of Counterparty Risk in Long Term Contracts Code Guide - Section 6

Michael Duarte Gonçalves

July 4, 2025

Overview

In this section, we extend our market modeling framework to incorporate the effects of public subsidies (T). The analysis builds upon the profit function introduced in the main paper, explicitly modeling how sellers' utilities and risk profiles are affected by the introduction of a per-unit subsidy.

Contents

1 Modeling Markets With Counterparty Risk (CPR) - Public Subsidies

1.1 Conceptual Approach

The workflow proceeds as follows:

1. **Parameterization and Simulation:** We define a range of subsidy values (T) and use functional programming and parallel processing to efficiently compute contract and spot market equilibria for each scenario. This includes solving for optimal forward positions and spot market participation for each plant, given their cost structures and risk aversion parameters.

2. **Market Equilibrium Computation:** For each combination of model parameters (γ, θ, T), we determine the market-clearing contract price and quantity. This is achieved by aggregating individual supply functions and matching them to demand, while accounting for the effect of subsidies on plant-level incentives.

3. **Welfare and Profit Analysis:** We compute welfare metrics under each scenario, decomposing total welfare into seller and buyer profits. The impact of subsidies is assessed both in absolute terms and relative to baseline (no-subsidy) outcomes. This allows for a detailed analysis of the distributional consequences of public subsidies mechanisms.

4. **Visualization and Export:** The results are visualized through a series of plots showing equilibrium prices, quantities, welfare, and profit shares as functions of the subsidy level and risk aversion. All key datasets are exported for further analysis and reproducibility.

1.2 Numerical Methods

1.2.1 Function `plot_profit_share_vs_T`

```r
plot_profit_share_vs_T <- function(data,
                                   y_var,
                                   y_label,
                                   file_name,
                                   save_path,
                                   y_scale_million = FALSE,
                                   y_scale_percent = FALSE,
                                   base_size = base_s,
                                   base_palette = base_palette,
                                   colors = NULL,
                                   legend_position = "bottom") {

  # Unique gamma groups
  gamma_levels <- sort(unique(data$gamma))
  n_groups <- length(gamma_levels)

  # Dynamic color assignment if not provided
  if (is.null(colors)) {
    colors <- gradient_n_pal(base_palette)(seq(0, 1, length.out = n_
        groups))
  }

  # Determine shapes
  filled_shapes <- 21:25
  available_shapes <- 0:25

  if (n_groups <= length(filled_shapes)) {
    shape_vals <- filled_shapes[1:n_groups]
    shape_mode <- "filled"
  } else {
    shape_vals <- available_shapes[1:n_groups]
    shape_mode <- "unfilled"
  }

  # Base plot
  plot <- ggplot(data, aes(
    x = T_values,
    y = .data[[y_var]],
    group = gamma,
    color = as.factor(gamma),
    shape = as.factor(gamma)
  )) +
    geom_line(linewidth = 1)

  # Add points
  if (shape_mode == "filled") {
    plot <- plot +
      geom_point(aes(fill = as.factor(gamma)), size = 2, stroke = 0.6)
          +
      scale_fill_manual(values = colors, name = expression(gamma))
  } else {
    plot <- plot + geom_point(size = 4)
  }
```

```
  # Scales and labels
  plot <- plot +
    scale_color_manual(values = colors, name = expression(gamma)) +
    scale_shape_manual(values = shape_vals, name = expression(gamma)) +
    labs(
      x = "T",
      y = y_label
    ) +
    theme_minimal(base_size = base_size) +
    theme(
      legend.position = legend_position,
      legend.title = element_text(face = "bold"),
      legend.box = if (identical(legend_position, "bottom")) "vertical"
          else NULL,
      panel.grid.major = element_line(color = "grey90", size = 0.2),
      panel.grid.minor = element_line(color = "grey95", size = 0.1)
    ) +
    guides(
      color = guide_legend(ncol = 1),
      shape = guide_legend(ncol = 1)
    )

  # Y-axis formatting
  if (y_scale_million) {
    plot <- plot + scale_y_continuous(
      labels = function(x) paste0(scales::comma(x / 1e6), "M")
    )
  }

  if (y_scale_percent) {
    plot <- plot + scale_y_continuous(
      labels = function(x) paste0(format(round(x, 1), nsmall = 0, trim
          = TRUE))
    )
  }

  # Show and save plot
  print(plot)

  ggsave(
    filename = file.path(save_path, file_name),
    plot     = plot,
    width    = 16,
    height   = 9,
    dpi      = 300
  )

  message("Saved: ", file_name)
}
```

- **Inputs:**
  - `data`: The data frame containing the results to plot.
  - `y_var`: The name of the variable to plot on the y-axis.
  - `y_label`: The label for the y-axis.
  - `file_name`: The name of the file to save the plot as.

  - `save_path`: The directory to save the plot.
  - `y_scale_million`: If `TRUE`, scales the y-axis in millions.
  - `y_scale_percent`: If `TRUE`, formats the y-axis as percentages.
  - `base_size`: Base font size for the plot.
  - `base_palette`: Color palette for the plot.
  - `colors`: Optional custom colors for the groups.
  - `legend_position`: Position of the legend in the plot.

- **Workflow:**
  1. Identifies the unique γ groups and assigns colors and shapes dynamically.
  2. Constructs a ggplot object with lines and points, grouping by γ.
  3. Applies custom color and shape scales.
  4. Optionally formats the y-axis in millions or as percentages.
  5. Adds labels, themes, and legend formatting.
  6. Prints and saves the plot to the specified file path.

- **Output:**
  - The function prints the plot to the screen and saves it as a PDF file in the specified directory.
  - A confirmation message indicates successful saving.

The function `plot_profit_share_vs_T` is an R function that we designed to visualize how a given variable (such as seller profits or profit shares) changes as a function of the public subsidy T, for different risk aversion groups (γ). The function is highly customizable, allowing the user to adjust axis formatting, color palettes, and legend positions.

1.2.2 Parallelized Scenario Expansion and Forward Position Calculation

This code chunk prepares the dataset for scenario analysis by expanding the data across all relevant parameter combinations and computing key forward market variables in parallel.

- **Parallel Processing Setup:** The code sets up parallel computation using `plan(multisession, workers = availableCores() - 1)`, enabling efficient evaluation of multiple scenarios simultaneously.
- **Scenario Expansion:** The base dataset (`wind_solar_proj_2022`) is expanded via a full crossing with all possible values of risk aversion (γ), subsidy levels (T), and demand (θ), creating a comprehensive grid of scenarios for analysis.
- **Forward Position Calculations:**
  - `f_c_cpr`: For each scenario, the optimal contract quantity in the CPR market is found using a root-finding function (`find_f_root`), with a default fallback value if the root cannot be found.
  - `f_spot_cpr`: Similarly, the optimal spot market quantity is found using `find_f_spot_root`, again with a default fallback.
  - `f_upper`: An upper bound for the forward position is computed for each scenario.
  - These computations are parallelized with `future_pmap_dbl` for speed.
- **Post-processing:**
  - The maximum feasible forward position (`f_max_cpr`) is determined for each scenario.
  - Additional variables are calculated, such as the product of contract price and quantity, and a message indicating whether the upper bound exceeds the expected price.
  - The data is arranged and cumulative sums of production and capacity are computed for each group.
  - Further variables relevant to welfare and subsidy calculations are added.

```
#  ---------------------------------------- #
#  ---------------------------------------- #

# Set your T values

# Set up parallel processing
plan(multisession, workers = availableCores() - 1)


# Split sample case
T_val <- seq(0, 0.05, by = 0.01)

# Improved version using functional programming principles
wind_solar_proj_2022_T <- wind_solar_proj_2022 %>%
  crossing(gamma = gamma_values,
           T_values = T_val,
           theta = theta_values) |>
  mutate(
    f_c_cpr = future_pmap_dbl(
      list(q_i_mwh, x, gamma, r_0, alpha, beta, total_cost, T_values),
      ~ coalesce(
```

```
        find_f_root(
          ..1, ..2, ..3, ..4, ..5, ..6, ..7, ..8
        ),
        1  # Default value
      )
    ),

    f_spot_cpr = future_pmap_dbl(
      list(q_i_mwh, x, gamma, r_0, alpha, beta, total_cost, expected_p,
          r, T_values),
      ~ coalesce(
        find_f_spot_root(
          ..1, ..2, ..3, ..4, ..5, ..6, ..7, ..8, ..9, ..10
        ),
        0  # Default value
      )
    ),
    f_upper = future_pmap_dbl(
      list(q_i_mwh, x, gamma, r_0, alpha, beta, total_cost, T_values),
      ~ find_upper_root(
          ..1, ..2, ..3, ..4, ..5, ..6, ..7, ..8
        )
      )
    ) %>%
  # Post-processing in vectorized operations
  mutate(
    f_max_cpr = pmax(f_c_cpr, f_spot_cpr, na.rm = TRUE),
    across(c(f_c_cpr, f_spot_cpr), ~ coalesce(., NA_real_))
  )

wind_solar_proj_2022_T <- wind_solar_proj_2022_T |>
  mutate(
    f_upper_message = if_else(
      !is.na(f_upper),
      if_else(f_upper > expected_p, "f_upper > E(p)", "f_upper <= E(p)"
          ),
      NA_character_),
    xf_c_cpr = x * f_c_cpr,
    xf_spot_cpr = x * f_spot_cpr,
    xf_upper_cpr = x * f_upper,
    xf_max_cpr = x * f_max_cpr
  )


wind_solar_proj_2022_T <- wind_solar_proj_2022_T |>
  arrange(theta, T_values, gamma, xf_max_cpr) |>
  group_by(theta, gamma, T_values) |>
  mutate(cumulative_production = cumsum(q_i_mwh),
         cumulative_capacity = cumsum(capacity)
         ) |>
  ungroup() |>
  mutate(x_q_exp_p_total_costs = x * expected_p * q_i_mwh - total_cost,
         lambda = lambda,
         lambda_q_i_mwh_T = lambda * T_values * q_i_mwh
  )
```

1.2.3 Contract Supply Curve for all Scenarios

This code block systematically generates and saves supply curve plots for each combination of subsidy level (T), share of opportunistic buyers (γ), and demand (θ) in the market simulation. The main steps are as follows:

1. **Iterate Over Subsidy Levels:** For each value of the public subsidy (T), the code:
   - Filters the main dataset to include only the current T value.
2. **Iterate Over Demand Scenarios:** For each unique demand value (θ) within the current T slice:
   - Filters the dataset further to the current (T, θ) scenario.
   - Converts the risk aversion parameter (γ) to a factor and numeric variable for plotting.
3. **Demand Line Construction:** A demand segment is created for the plot, representing the demand at the current θ and threshold price.
4. **Supply Curve Plotting:**
   - Uses `ggplot2` to plot the supply curve as a step function of cumulative capacity vs. contract price, colored by γ.
   - Adds points for each plant and overlays the demand line.
   - Applies custom color gradients, axis labels, and theming for clarity.
5. **File Naming and Saving:**
   - Generates a descriptive filename based on the current T and θ.
   - Saves the resulting plot as a PDF to the specified directory.
   - Prints a confirmation message for each saved plot.

This process results in a comprehensive set of supply function plots, one for each scenario, which are essential for visualizing how contract supply responds to changes in subsidies and demand.

```r
for (i in seq_along(T_val)) {
  T_value <- T_val[i]

  # Filter for current T slice
  wind_T_data <- wind_solar_proj_2022_T |>
    filter(T_values == T_value)

  # Get unique theta values
  theta_values <- sort(unique(wind_T_data$theta))

  for (j in seq_along(theta_values)) {
    theta_val <- theta_values[j]

    # Filter for this (T, theta)
    slice_data <- wind_T_data |>
      filter(theta == theta_val) |>
      mutate(
        gamma = as.factor(gamma),
        gamma_num = as.numeric(as.character(gamma))
```

```
  )

# Demand line
demand_segments <- tibble(
  x_start = theta_val,
  x_end   = theta_val,
  y_start = threshold_price,
  y_end   = min(slice_data$xf_max_cpr, na.rm = TRUE)
)

# Plot
supply_plot <- ggplot(
  slice_data |> filter(xf_max_cpr <= expected_p * x),
  aes(
    x = cumulative_capacity,
    y = xf_max_cpr,
    group = gamma,
    color = gamma_num
  )
) +
  geom_step() +
  geom_point(size = 0.5) +
  geom_segment(
    data = demand_segments,
    aes(x = x_start, xend = x_end, y = y_start, yend = y_end),
    color = "black", linetype = "solid", size = 1,
    inherit.aes = FALSE
  ) +
  scale_color_gradientn(
    colours = theme_palette_gamma,
    name = expression(gamma)
  ) +
  labs(
    x = "Cumulative Capacity (MW)",
    y = "Contract Price (EUR/MWh)"
  ) +
  theme_minimal(base_size = base_s) +
  theme(
    legend.position = c(0.05, 0.95),
    legend.justification = c(0, 1),
    legend.background = element_rect(fill = alpha("white", 0.2),
       color = NA),
    legend.title = element_text(face = "bold"),
    panel.grid.major = element_line(color = "grey90", size = 0.2),
    panel.grid.minor = element_line(color = "grey95", size = 0.1)
  )

# Labels for filename
theta_label <- format(round(theta_val), big.mark = "")
T_label <- ifelse(
  T_value < 1,
  formatC(T_value, format = "f", digits = 3),
  as.character(T_value)
)

filename <- paste0(
  sprintf("%02d", i),
  "_supply_function_cpr_T_", T_label,
```

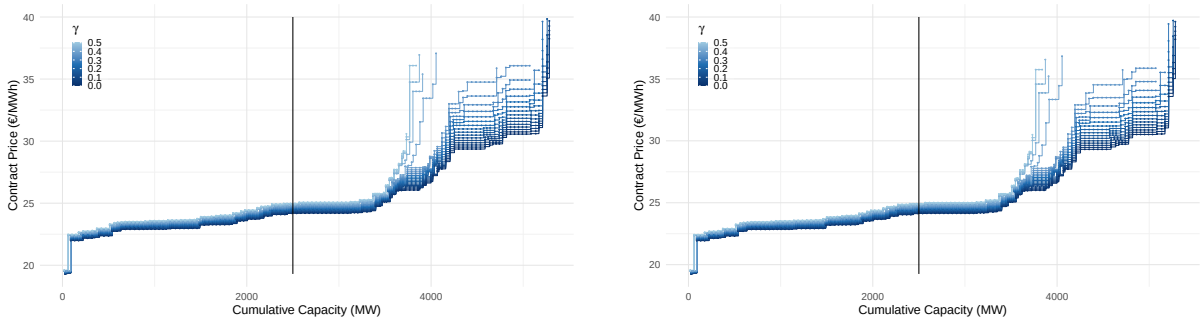
```
      "_theta_", theta_label, ".pdf"
    )

    plot_path <- file.path(with_T_fig_path, filename)

    ggsave(plot_path, plot = supply_plot, width = 16, height = 9, dpi =
        300)

    message("Saved plot for T = ", T_label, ", theta = ", theta_label,
       " at: ", plot_path)
  }
}
```

Figure 1 represents some contract supplies generated with the above code.



(a) $\theta = 2500, T = 0$ (b) $\theta = 2500, T = 0.05$

Figure 1: Contract Supplies with Public Subsidies T

1.2.4 Equilibrium Price and Quantity Computation

This code block computes the contract market equilibrium price and quantity for each scenario defined by risk aversion (γ), subsidy level (T), and demand (θ). The process is as follows:

1. **Identify the Last Seller Before Demand is Met:**
    - For each group (γ, T, θ), the code sorts plants by cumulative capacity.
    - It filters for all plants whose cumulative capacity is *less than* the demand (θ), and whose price offer is below the threshold price.
    - It selects the last such plant (i.e., the one just before the demand is reached).
2. **Peek at the Next Seller (First After Demand is Met):**
    - Again, for each group, it finds the first plant whose cumulative capacity is *greater than or equal to* the demand (θ).
    - This gives the price that would be set if the demand just crosses into the next plant's offer.
3. **Join and Determine the Equilibrium Price:**
    - The two datasets are joined to align the "before" and "after" rows for each scenario.
    - The equilibrium price is set as follows:
    - The equilibrium quantity is set as the cumulative capacity just before the demand is met.

4. **Result:**
    - The output is a table (`equilibrium_prices_T`) giving, for each scenario, the equilibrium quantity and price that clear the contract market.
    - This table is sorted for clarity and printed for inspection.

1.2.5 Integration of Equilibrium Results and Calculation of Plant-Level Metrics

This section of the code integrates the equilibrium contract price and quantity into the plant-level scenario dataset and computes additional variables relevant for welfare and profit analysis.

```
# Equilibrium Prices Dataset + Some reordering

wind_solar_proj_2022_T <- wind_solar_proj_2022_T |>
  left_join(
    equilibrium_prices_T |>
      select(gamma, theta, T_values, equilibrium_price, equilibrium_
          quantity),
    by = c("gamma", "T_values", "theta")
  )

wind_solar_proj_2022_T <- wind_solar_proj_2022_T |>
  mutate(f_equilibrium = equilibrium_price / x,
         xf_equilibrium = equilibrium_price) |>
  select(-equilibrium_price)


wind_solar_proj_2022_T <- wind_solar_proj_2022_T |>
  arrange(theta, T_values, gamma, xf_max_cpr) |>
  rowwise() |>
  mutate(R_f_equilibrium_cpr =
           compute_R_value_gamma(f_equilibrium, gamma, q_i_mwh, x, r_0,
               alpha, beta)) |>
  mutate(
    x_q_exp_p_total_costs_R = x * expected_p * q_i_mwh - total_cost
    - R_f_equilibrium_cpr
  )

wind_solar_proj_2022_T <- wind_solar_proj_2022_T |>
  mutate(
    profit_cpr_contracts =
      vectorized_pi_s(
        f = f_equilibrium,
        q_i = q_i_mwh,
        x = x,
        gamma = gamma,
        r_0 = r_0,
        alpha = alpha,
        beta = beta,
        total_costs = total_cost,
        T_values = T_values
      )
  )


ordered_vars <- c(
```

```
  "theta", "gamma", "lambda", "lambda_q_i_mwh_T", "T_values", "f_c_cpr"
     , "f_spot_cpr",
  "f_upper", "f_upper_message", "f_max_cpr", "f_equilibrium",
  "R_f_equilibrium_cpr", "xf_c_cpr", "xf_spot_cpr", "xf_upper_cpr", "xf
     _max_cpr",
  "xf_equilibrium", "equilibrium_quantity", "x_q_exp_p_total_costs", "x
     _q_exp_p_total_costs_R",
  "cumulative_production", "cumulative_capacity", "profit_cpr_contracts
     "
)

# Reorder dataframe
wind_solar_proj_2022_T <- wind_solar_proj_2022_T |>
  select(
    everything(),               # Keeps all variables in current order
       ...
    -all_of(ordered_vars),      # ...but temporarily removes the ones
       you want to reorder
    -profits_sp_no_cpr,                 # temporarily remove profits_sp_
       no_cpr to place it explicitly
    profits_sp_no_cpr,                  # put profits_sp_no_cpr in place
    all_of(ordered_vars)        # then add ordered vars in the desired
       order
  )
```

The main steps are as follows:

1. **Merging Equilibrium Results:**
   - For each plant i and scenario (γ, T, θ), the equilibrium contract price f^* and equilibrium quantity q^* (computed previously) are merged into the main dataset.
2. **Calculation of Derived Variables:**
   - The f^* is given by:
   $$f^* = \frac{xf^*}{x} \equiv \frac{\texttt{equilibrium\_price}}{x}$$
   we then rename `equilibrium_price` (which is the same as xf^* to `xf_equilibrium` to be consistent to previous names.
   - For each plant, we compute the $R(f^*, \gamma)$.
   - We also compute for each plant i:
   $$\mathbb{E}(p) \cdot xq_i - C(k_i) - R(f^*, \gamma)$$
   - For each combination of γ, θ, and T we compute seller profits under contracts as

$$\pi_i^S \equiv q_i x \left[\gamma \int_0^{f^*} p\,\phi(p)\,dp + f^* \cdot \Big(1 - \gamma\Phi(f^*)\Big)\right] - R_i(f^*, \gamma) - C(k_i) + T\,q_i.$$

⚠ **Reminder**: Now, the profit function contains the last term $T\,q_i$ that is not fixed necessarily to 0! The function `vectorized_pi_s` was defined in Section 4.

- **Reordering Variables:**
  - The columns in the dataset are reordered so that all key variables (such as θ, γ, T, equilibrium prices and quantities, profits, and risk adjustments) are grouped together in a logical sequence, facilitating further analysis and export.

1.2.6 Welfare Dataset Creation and Analysis

```
# Welfare Dataset Creation

# STEP 1: W_0 (welfare under gamma = 0, profits spot > 0)
W_0_T <- wind_solar_proj_2022_T |>
  filter(gamma == 0, profits_sp_no_cpr >= 0) |>
  group_by(T_values, theta) |>
  summarise(
    welfare_0_eur = sum(x_q_exp_p_total_costs - r, na.rm = TRUE),
    .groups = "drop"
  ) |>
  ungroup()

# STEP 2: W_gamma for all gamma-theta combinations (adjusted for
    guarantees)
W_gamma_T <- wind_solar_proj_2022_T |>
  group_by(gamma, T_values, theta) |>
  filter(xf_max_cpr <= xf_equilibrium & cumulative_capacity < theta) |>
  summarise(
    x_q_exp_p_total_costs_R = sum(
      x_q_exp_p_total_costs_R
    ),
    lambda_T_q = sum(lambda * T_values * q_i_mwh),
    welfare_gamma_eur = x_q_exp_p_total_costs_R - lambda_T_q,
    .groups = "drop",
    theta = first(theta)
  ) |>
  ungroup()


# STEP 3: Baseline gamma = 0 values for comparison
W_gamma_0_T <- W_gamma_T |>
  group_by(theta, T_values) |>
  filter(gamma == 0) |>
  select(theta, gamma, T_values, welfare_gamma_0_eur = welfare_gamma_
      eur) |>
  ungroup()


# STEP 4: Equilibrium quantities/prices at gamma = 0 (for ratio
    comparisons)
eq_gamma_0_T <- equilibrium_prices_T |>
  group_by(theta, T_values) |>
  filter(gamma == 0) |>
  select(
    theta,
    gamma,
    T_values,
    equilibrium_quantity_gamma_0 = equilibrium_quantity,
    equilibrium_price_gamma_0 = equilibrium_price
  ) |>
    ungroup()



# Combine all metrics into one final table
```

```
welfare_dataset_T_full <- equilibrium_prices_T |>
  left_join(W_gamma_T, by = c("theta", "gamma", "T_values")) |>

  left_join(
    W_gamma_0_T |> select(theta, T_values, welfare_gamma_0_eur),
    by = c("theta", "T_values")
  ) |>

  left_join(W_0_T, by = c("theta", "T_values")) |>

  left_join(eq_gamma_0_T |> select(-gamma), by = c("theta", "T_values")
     ) |>

  mutate(
    equilibrium_price_eur     = equilibrium_price,
    equilibrium_quantity_mw   = equilibrium_quantity,

    W_0 = if_else(gamma == 0, welfare_0_eur, NA_real_),

    welfare_gain_vs_baseline_meur = (welfare_gamma_eur - welfare_0_eur)
        / 1e6,
    welfare_ratio_percent         = (welfare_gamma_eur / welfare_gamma_
       0_eur) * 100,
    welfare_gap_million_eur       = (welfare_gamma_0_eur - welfare_
       gamma_eur) / 1e6,

    eq_quantity_ratio_percent = (equilibrium_quantity / equilibrium_
       quantity_gamma_0) * 100,
    eq_price_ratio_percent    = (equilibrium_price / equilibrium_price_
       gamma_0) * 100
  ) |>

  select(
    theta, gamma, T_values,
    equilibrium_price_eur,
    eq_price_ratio_percent,
    equilibrium_quantity_mw,
    eq_quantity_ratio_percent,
    W_0,
    welfare_gamma_eur,
    welfare_ratio_percent,
    welfare_gap_million_eur,
    welfare_gain_vs_baseline_meur
  ) |>

  arrange(T_values, gamma)
```

This code block constructs a comprehensive welfare dataset by calculating welfare metrics under different risk aversion levels (γ), subsidy levels (T), and demand scenarios (θ). The steps are as follows:

1. **Welfare W^0 Calculation:**
    - For the baseline case where share of opportunistic buyers $\gamma = 0$ and spot market profits are non-negative, welfare is computed as:

$$W^0(T, \theta) = \sum_i (xq_i \mathbb{E}(p) - C(k_i) - r_i)$$

2. **Welfare for $W(\gamma)$ Calculation:**

   - For each combination of T, θ, welfare is computed considering risk adjustments and subsidies for winning plants as:

$$W(\gamma) = \sum_i (xq_i \mathbb{E}(p) - C(k_i) - R_i(f^*, \gamma)) - \lambda T \sum_i q_i$$

     where $R_i(f^*, \gamma)$ is the risk adjustment, $\lambda = 0.3$ is fixed and can be interpreted as the per-unit cost of social funds, and T is the subsidy.

3. **Welfare for Comparison:**

   - Extracts welfare values for $\gamma = 0$ to serve for different calculations.

4. **Equilibrium Quantities and Prices when $\gamma = 0$**

   - Extracts equilibrium quantities and prices for $\gamma = 0$ to enable ratio comparisons.

5. **Combining Metrics into a Final Dataset:**

   - Joins all computed metrics into one dataset.
   - Calculates additional variables:

$$\text{welfare\_gain\_vs\_baseline\_meur} = \frac{W(\gamma) - W^0}{10^6}$$

$$\text{welfare\_ratio\_percent} = \frac{W(\gamma)}{W(\gamma = 0)} \times 100$$

$$\text{welfare\_gap\_million\_eur} = \frac{W(\gamma = 0) - W(\gamma)}{10^6}$$

$$\text{eq\_quantity\_ratio\_percent} = \frac{q^*}{q^*_{\gamma=0}} \times 100$$

$$\text{eq\_price\_ratio\_percent} = \frac{f^*_\gamma}{f^*_{\gamma=0}} \times 100$$

6. **Final Dataset Arrangement:**

   - Selects and orders relevant columns for clarity.
   - Sorts the dataset by subsidy T and risk aversion γ.

1.2.7 Welfare Ratio Analysis at Fixed Demand

```
# for subsequent graphs/tables, we only loot at theta = 3,500
# Compute W(gamma) for each gamma

welfare_ratios_filtered <- W_gamma_T |>
  filter(theta == selected_theta) |>
  left_join(welfare_dataset_baseline, by = c("gamma", "theta")) |>
  mutate(
    welfare_ratio = (welfare_gamma_eur / welfare_gamma_eur_baseline) *
       100
    ) |>
  select(theta, gamma, T_values, everything()) |>
  arrange(gamma, T_values)

welfare_ratios_wide_T <- welfare_ratios_filtered |>
  select(T_values, gamma, theta, welfare_ratio) |>
  pivot_wider(
    names_from = T_values,
    values_from = welfare_ratio,
    names_prefix = "T_val_"
  )
```

This code block narrows the welfare analysis to a single demand scenario ($\theta = 3500$), and for $\gamma \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$. It computes, for each share of opportunistic buyers (γ) and subsidy (T), the welfare ratio relative to the baseline computed at Section 4.

1. **Filter for Selected Demand:**
   - The dataset is filtered to only include records where $\theta = 3500$.
2. **Join with Baseline Welfare:**
   - For each γ, the welfare under the current scenario (welfare_gamma_eur) is joined with the corresponding baseline welfare (welfare_gamma_eur_baseline), enabling direct comparison.
3. **Compute Welfare Ratio:**
   - The welfare ratio is calculated as:

$$\text{welfare\_ratio} = \frac{W(\gamma)^T}{W(\gamma)^{Baseline}} \times 100$$

     where $W(\gamma)^T$ is the welfare for a given γ in the public subsidies case, and $W(\gamma)^{Baseline}$ is the baseline welfare for the same γ.
4. **Reshape for Presentation:**
   - The resulting data is reshaped into a wide format, so that each column corresponds to a different value of the subsidy T, making it easier to compare welfare ratios across T for each γ.

1.2.8 Profit Decomposition and Share Calculation under Subsidy T

```
# Profits under T

profits_by_gamma_theta_T <- wind_solar_proj_2022_T |>
  group_by(theta, gamma, T_values) |>
  filter(xf_max_cpr <= xf_equilibrium & cumulative_capacity < theta) |>
  summarise(
    seller_profits_eur_T = sum(profit_cpr_contracts, na.rm = TRUE),
    .groups = "drop",
    theta = first(theta)
  ) |>
  ungroup() |>
  arrange(theta, gamma, T_values, seller_profits_eur_T)


welfare_T_with_profits <- welfare_dataset_T_full |>
  left_join(profits_by_gamma_theta_T, by = c("gamma", "theta", "T_
     values"))

# Step 3: Compute profit shares for sellers and buyers
welfare_T_with_profits <- welfare_T_with_profits |>
  mutate(
    buyer_profits_eur_T = welfare_gamma_eur - seller_profits_eur_T,
    seller_profit_T_share_percent = round((seller_profits_eur_T /
       welfare_gamma_eur) * 100, 2),
    buyer_profit_T_share_percent  = round((buyer_profits_eur_T /
       welfare_gamma_eur) * 100, 2)
  ) |>
  select(
    gamma, theta, T_values,
    welfare_gamma_eur,
    seller_profits_eur_T,
    buyer_profits_eur_T,
    seller_profit_T_share_percent,
    buyer_profit_T_share_percent
  ) |>
  arrange(theta, gamma)
```

This code block calculates the distribution of welfare between sellers and buyers for each scenario, focusing on the effect of the subsidy T. The steps are as follows:

1. **Aggregate Seller Profits:**
   - For each combination of demand (θ), risk aversion (γ), and subsidy (T), sum the contract profits across all the winning plants i:

     $$\Pi^S(f^*) = \sum_i \pi_i^S$$

     where π_i^S is the profit from contracts for plant i, and the sum is over plants with $xf_{\max} \leq xf^*$ and cumulative capacity less than θ.
2. **Merge with Welfare Data:**
   - Join the seller profits with the full welfare dataset for each scenario, so that total welfare and seller profits are available together.
3. **Compute Buyer Profits and Shares:**

- Calculate buyer profits as the residual, for each combination of demand (θ), risk aversion (γ), and subsidy (T):

$$\Pi^{Buyers} = W(\gamma) - \Pi^{S}(f^*)$$

- Compute the profit shares (in percent) for sellers and buyers:

$$\text{Seller Share}_{\gamma,\theta,T} = \frac{\Pi^{S}(f^*)}{W(\gamma)} \times 100$$

$$\text{Buyer Share}_{\gamma,\theta,T} = \frac{\Pi^{Buyers}}{W(\gamma)} \times 100$$

1.2.9 Visualization of Market Outcomes Across Subsidy Levels

```
# Plots

# We select a fewer gammas and only for selected_theta
# to avoid multiple lines in plots

# Filter datasets
equilibrium_prices_T_filtered <- equilibrium_prices_T |>
  filter(gamma %in% selected_gammas,
         theta %in% selected_theta)

welfare_dataset_T_filtered <- welfare_dataset_T_full |>
  filter(gamma %in% selected_gammas,
         theta %in% selected_theta)

welfare_T_with_profits_filtered <- welfare_T_with_profits |>
  filter(gamma %in% selected_gammas,
         theta %in% selected_theta)


wind_solar_proj_2022_T_filtered <- wind_solar_proj_2022_T |>
  filter(gamma %in% selected_gammas,
         theta %in% selected_theta)


theme_palette_gamma_selected <- gradient_n_pal(base_palette)(
  seq(0, 1, length.out = length(selected_gammas))
)

# Prices
plot_eq_price <- plot_line_by_gamma(
  data         = equilibrium_prices_T_filtered,
  x            = "T_values",
  y            = "equilibrium_price",
  group_var    = "gamma",
  color_var    = "gamma",
  color_label  = expression(gamma),
  x_lab        = expression(T),
  y_lab        = "Contract Price (EUR/MW)",
  colors       = theme_palette_gamma_selected,
  filename     = "06_equilibrium_price_vs_gamma_T.pdf",
  folder       = with_T_fig_path,
  legend_position = c(0.05, 0.70)
```

```
)

# Quantities
plot_eq_quantities <- plot_line_by_gamma(
  data           = equilibrium_prices_T_filtered,
  x              = "T_values",
  y              = "equilibrium_quantity",
  group_var      = "gamma",
  color_var      = "gamma",
  color_label    = expression(gamma),
  x_lab          = expression(T),
  y_lab          = "Investment (MW)",
  y_scale_comma  = TRUE,
  colors         = theme_palette_gamma_selected,
  filename       = "07_equilibrium_quantity_vs_gamma_T.pdf",
  folder         = with_T_fig_path,
  legend_position = c(0.05, 0.70)
)

# Welfare
plot_welfare <- plot_line_by_gamma(
  data             = welfare_dataset_T_filtered,
  x                = "T_values",
  y                = "welfare_gamma_eur",
  group_var        = "gamma",
  color_var        = "gamma",
  color_label      = expression(gamma),
  x_lab            = expression(T),
  y_lab            = "Welfare (MEUR)",
  y_scale_million  = TRUE,
  colors           = theme_palette_gamma_selected,
  filename         = "08_equilibrium_welfare_vs_gamma_T.pdf",
  folder           = with_T_fig_path,
  legend_position  = c(0.05, 0.30)
)

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

# Profits Plots

plot_profit_share_vs_T(
  data              = welfare_T_with_profits_filtered,
  y_var             = "seller_profits_eur_T",
  y_label           = "Seller Profit (MEUR)",
  file_name         = "09_seller_profit_vs_gamma_T.pdf",
  save_path         = with_T_fig_path,
  y_scale_million   = TRUE,
  base_palette      = base_palette,
  legend_position = c(0.05, 0.85)
)


plot_profit_share_vs_T(
  data              = welfare_T_with_profits_filtered,
  y_var             = "seller_profit_T_share_percent",
  y_label           = "Seller Profit (%)",
  file_name         = "10_seller_profit_share_vs_gamma_T.pdf",
  save_path         = with_T_fig_path,
```

```
  y_scale_percent  = TRUE,
  base_palette     = base_palette,
  legend_position = c(0.05, 0.85)
)
```

This code block generates a series of plots to visualize how key market outcomes—equilibrium prices, investment quantities, welfare, and profit shares—vary as functions of the subsidy level T, focusing on selected risk aversion levels (γ) and a fixed demand (θ). The process is as follows:

1. **Data Filtering:**
   - All relevant datasets are filtered to include only the selected values of γ and the chosen θ, ensuring that the plots remain clear and interpretable.
2. **Color Palette Construction:**
   - A color palette is generated for the selected γ values using a gradient based on the base color palette, ensuring visual consistency across plots.
3. **Plotting Equilibrium Prices and Quantities:**
   - The function `plot_line_by_gamma`, which was created in Section 4, is used to plot equilibrium contract prices and investment quantities as functions of the subsidy T, with lines grouped and colored by γ.
4. **Plotting Welfare:**
   - Welfare is plotted similarly, with the y-axis scaled in millions for readability.
5. **Plotting Profits:**
   - The function `plot_profit_share_vs_T`, defined at the beginning of this section, is used to plot seller profits (both in euros and as a percentage of welfare) as functions of T.
6. **File Saving and Legend Positioning:**
   - Each plot is saved to disk with a descriptive filename, and legends are positioned for optimal clarity.

Note:
The plotting functions `plot_line_by_gamma` and `plot_profit_share_vs_T` were developed in Section 4 and in Section 6, respectively.

1.2.10 Generated Figures

Table 1: Summary of Figures Generated for Public Subsidies Analysis

# Plot	Description
# 06	Equilibrium contract price as a function of T and γ
# 07	Equilibrium investment (MW) as a function of T and γ
# 08	Total welfare (in million €) as a function of T and γ
# 09	Seller profit (in million €) as a function of T and γ
# 10	Seller profit share (%) as a function of T and γ

1.2.11 Exporting Public Subsidies Results

```
# Save everything: for now, we save only results for theta = 3500

# Define dataset list for T-based results
datasets_T <- list(
  `Wind Solar Projects (T)` = wind_solar_proj_2022_T_filtered,
  `Equilibrium P. & Q.`     = equilibrium_prices_T_filtered,
  `Welfare with T`          = welfare_dataset_T_filtered,
  `Welfare Ratios`          = welfare_ratios_filtered,
  `Profits (with T)`        = welfare_T_with_profits_filtered
)

# Filter all datasets by gamma
filtered_T <- lapply(datasets_T, filter_by_gamma)

# Label theta for filename
theta_label <- format(round(unique(wind_solar_proj_2022_T_filtered$
    theta)), big.mark = "")

# Construct output path
excel_filename_T <- paste0("01_wind_solar_projects_cpr_T_theta_", theta
    _label, ".xlsx")
output_path_T <- file.path(with_T_path, excel_filename_T)

# Create workbook
wb_T <- createWorkbook()

# Add sheets dynamically
for (sheet in names(filtered_T)) {
  addWorksheet(wb_T, sheet)
  freezePane(wb_T, sheet, firstActiveRow = 2, firstActiveCol = 2)
  writeData(wb_T, sheet = sheet, x = filtered_T[[sheet]])
}

# Save workbook
saveWorkbook(wb_T, file = output_path_T, overwrite = TRUE)
```

This final code block consolidates and exports all key scenario results for the selected demand level ($\theta = 3500$) into a single Excel workbook, facilitating further analysis and reproducibility. The process is as follows:

1. **Dataset Collection:**
    - A list of all relevant filtered datasets is created, including:
        - Plant-level scenario results (`Wind Solar Projects (T)`)
        - Equilibrium prices and quantities (`Equilibrium P. & Q.`)
        - Welfare metrics (`Welfare with T`)
        - Welfare ratios (`Welfare Ratios`)
        - Profits (`Profits (with T)`)
2. **Filtering by Share of Opportunistic Buyers:**
    - All datasets are further filtered by the selected set of risk aversion levels (γ), ensuring only relevant scenarios are included.
3. **File Naming and Path Construction:**

- The value of θ is extracted and formatted for use in the output filename.
- The output path for the Excel file is constructed accordingly.

4. **Workbook Creation and Sheet Writing:**
   - An Excel workbook object is created.
   - For each dataset, a new worksheet is added, the data is written, and the first row and column are frozen for easier navigation.
5. **Saving the Workbook:**
   - The workbook is saved to disk at the specified location, with overwriting enabled to ensure the latest results are always available.

1.2.12 Created Tables

Table 2: Variables in `Wind Solar Projects (T)` Sheet

Variable	Description
projectname	Project name
capacity	Installed capacity (MW)
avgcapacityfactor	Average capacity factor
type	Technology type (Wind/Solar)
hours	Annual full-load hours
power_kw	Installed power (kW)
q_i_kwh	Annual generation (kWh)
q_i_mwh	Annual generation (MWh)
v_q_i_mwh	Consumer valuation (MWh)
c_inv	Investment cost per MW
c_om	O&M cost per MW
total_cost	$C(k_i)$
avg_cost_euro_kwh	Average cost (EUR/kWh)
avg_cost_euro_mwh	Average cost (EUR/MWh)
r_0	r_0
r	r_i
profits_sp_no_cpr	Π_S^0
theta	Demand scenario θ (MW)
gamma	Share of opportunistic buyers γ
lambda	Per-unit Cost of Social Funds λ
lambda_q_i_mwh_T	$\lambda \cdot q_i \cdot T$
f_c_cpr	f_c
f_spot_cpr	f_{spot}
f_upper	Second root f_{upper}, if any
f_upper_message	Says if `f_upper`$> \mathbb{E}(p)$ or `f_upper`$\leq \mathbb{E}(p)$
f_max_cpr	f_{max}
f_equilibrium	f^*
R_f_equilibrium_cpr	$R_i(f^*, \gamma)$
xf_c_cpr	xf_c
xf_spot_cpr	xf_{spot}
xf_upper	Second root xf_{upper}, if any
xf_max_cpr	xf_{max}
xf_equilibrium	xf^*
equilibrium_quantity	q^*
x_q_exp_p_total_costs	$xq_i \cdot \mathbb{E}(p) - C(k_i)$
x_q_exp_p_total_costs_R	$xq_i \cdot \mathbb{E}(p) - C(k_i) - R_i(f^*, \gamma)$
cumulative_production	Cumulative energy production (MWh)
cumulative_capacity	Cumulative capacity (MW)
profit_cpr_contracts	Profits under CPR contracts, using f^*

Table 3: Variables in `Equilibrium P. & Q.` Sheet

Variable	Description
gamma	Share of opportunistic buyers γ
theta	Demand scenario (MW) - θ
T_values	Public Subsidies Values T
equilibrium_quantity	Total contracted quantity at equilibrium (MW), q^*
equilibrium_price	Equilibrium contract price xf^* (EUR/MWh)

Table 4: Variables in `Welfare with T` Sheet

Variable	Description
gamma	Share of opportunistic buyers γ
theta	Demand scenario (MW) - θ
T_values	Public Subsidies Values T
equilibrium_price_eur	Equilibrium contract price xf^* (EUR/MWh)
eq_price_ratio_percent	Price as % of $\gamma = 0$ baseline
equilibrium_quantity_mw	Equilibrium contracted quantity (MW), q^*
eq_quantity_ratio_percent	Quantity as % of $\gamma = 0$ baseline
W_0	Welfare W^0
welfare_gamma_eur	Welfare $W(\gamma)$ for given θ (EUR)
welfare_ratio_percent	Welfare as % of $W(\gamma = 0)$ baseline
welfare_gap_million_eur	Welfare gap vs. baseline (million EUR)
welfare_gain_vs_baseline_meur	Welfare gain vs. baseline (million EUR)

Table 5: Variables in `Welfare Ratios` Sheet

Variable	Description
theta	Demand scenario (MW) - θ
gamma	Share of opportunistic buyers γ
T_values	Public Subsidies Values T
x_q_exp_p_total_costs_R	$\sum_i xq_i \cdot \mathbb{E}(p) - C(k_i) - R_i(f^*, \gamma)$, for each (γ, θ, T) combinations
lambda_T_q	$\lambda T \sum_i q_i$ for each (γ, θ, T) combinations
welfare_gamma_eur	Welfare (EUR), with public Subsidies T
welfare_gamma_eur_baseline	Baseline welfare (EUR), no public guarantees (*c.f.* Section 4)

Table 6: Variables in `Profits (with T)` Sheet

Variable	Description
gamma	Share of opportunistic buyers γ
theta	Demand scenario (MW) - θ
T_values	Public Subsidies Values T
welfare_gamma_eur	Welfare (EUR), with public Subsidies T
seller_profits_eur_T	Seller profits under public subsidies T (EUR)
buyer_profits_eur_T	Buyer profits under public subsidies T (EUR)
seller_profit_T_share_percent	Seller share of welfare (%)
buyer_profit_T_share_percent	Buyer share of welfare (%)