

Supplementary Material 1

LCC system prototype functions 1-3
(+ plain language code)

Function 1

Asset management decision support software (beta)

Total-cost-of-ownership module

Study scope

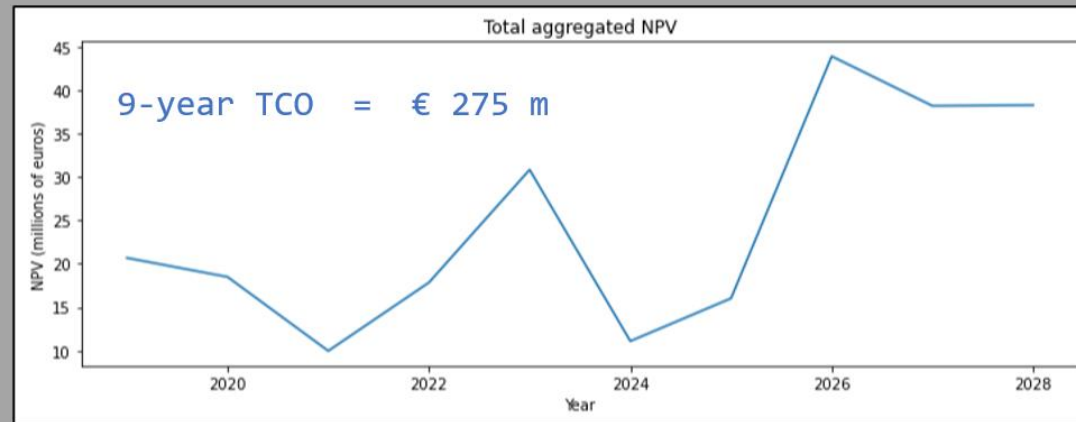
study_period_start	study_period_end	discount_rate	inflation_rate
2019	2028	0.0416	0.02

Asset register - input data

asset_id	installation_date	lifetime	initial_cost	replacement_cost	annual_energy	annual_maintenance	renovation_cost	renovation_frequency
200439	2017	25	100000	100000	130.00	2081.50	5000	7
202340	2005	25	50000	50000	11.70	495.43	2500	4
9657	1993	25	800000	800000	7884.00	8050.19	8000	4
9658	1993	25	800000	800000	7884.00	8050.19	8000	4
15770	2010	10	20000	20000	1664.40	3676.38	2500	5
15771	2000	10	20000	20000	1664.40	3676.38	2500	5
15777	2000	10	20000	20000	1664.40	3676.38	2500	5
15779	2000	10	20000	20000	1664.40	3676.38	2500	5



inputs



display

This function enables the user to calculate the total-cost-of-ownership across a portfolio of assets. The demonstration above aggregates life cycle cost calculations for approx. 4000 assets and displays a cashflow chart over 9 years.

Function 2

Asset management decision support software (beta)

Strategy comparison module

Study scope

study_period_start: 2019
study_period_end: 2094
study_period: 75
discount_rate: 0.0416
inflation_rate: 0.02

Option A - input data

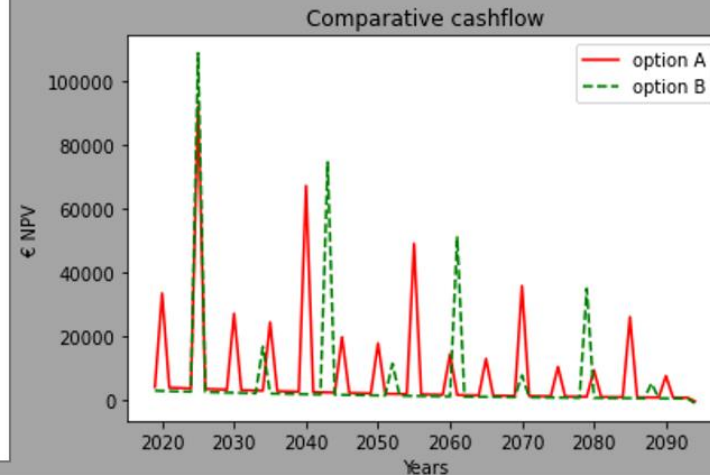
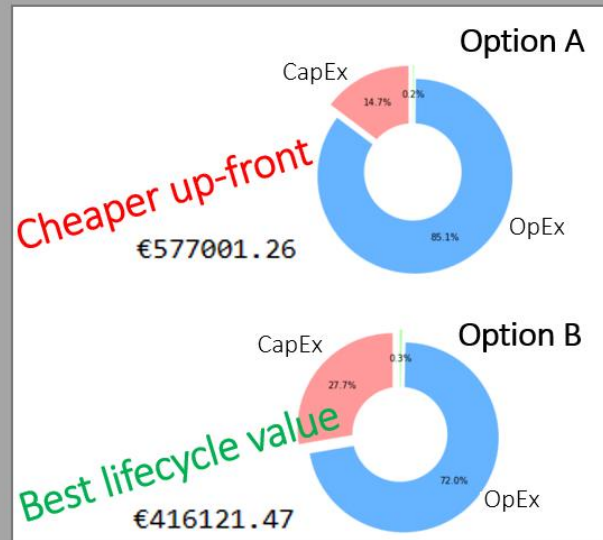
asset_id: 82344416
asset_type: Noodstroom
installation_date: 2010
lifetime: 15
initial_cost: 100000.0
replacement_cost: 100000.0
residual_value: 5000
annual_energy: 219.0
annual_maintenance: 4016
renovation_cost: 30000.0
renovation_frequency: 5

Option B - input data

asset_id: 82344416_B
asset_type: Noodstroom
installation_date: 2025
lifetime: 18
initial_cost: 120000
replacement_cost: 120000
residual_value: 7000
annual_energy: 100
annual_maintenance: 3000
renovation_cost: 20000
renovation_frequency: 9



inputs



display

This function compares life cycle cost analysis between two options on an individual asset level, displaying cashflow and phase-breakdown charts. In the above example a more expensive upfront investment is cheaper over the 75-year facility lifetime.

Function 3

Asset management decision support software (beta)

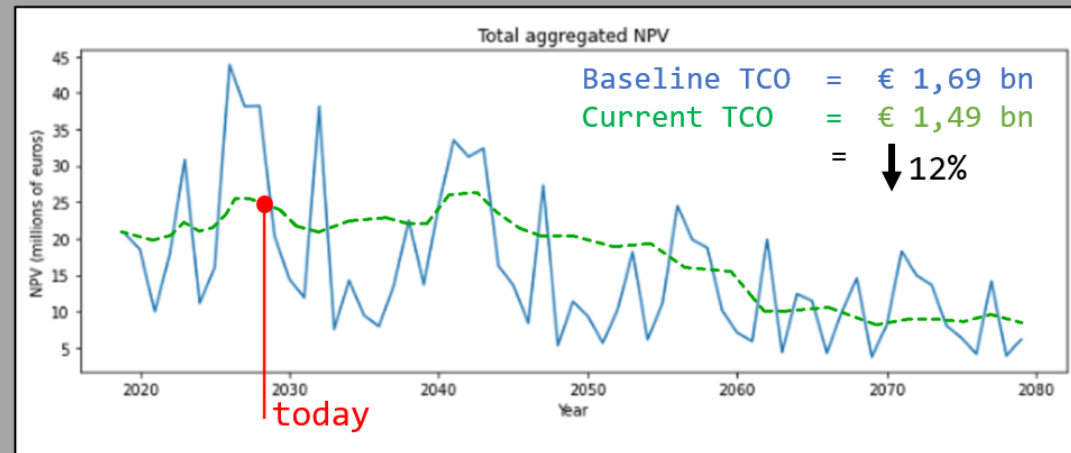
Baseline comparison module

Study scope

baseline_year : 2019
measurement_year : 2028
study_period (yr) : 60



inputs



display

This function compares a predicted (baseline) life cycle cost calculation and the measured performance during asset management, or the *performance gap*. Such functionality will be necessary to support performance-based contracting in future.

Plan language software code

Function 1 – Aggregated TCO

- A) F1 - LCC analysis function
- Build dataframe with years based on 'study period'
 - Add columns for single cost inputs ('acquisition', 'residual' etc.) and specify when occurring ('acquisition' at 'installation date', 'residual' end of 'study period')
 - Add columns for annual cost inputs (energy, maintenance etc.)
 - Add columns for variable cost inputs and apply a 'frequency' starting from 'installation date' (renovation every 'lifetime' years)
 - Sum all cost columns into new 'total' column
 - Add a column 'total+infl.' and multiply 'total' column by 'inflation rate'
 - Add column 'NPV' and multiply 'total+inflation' by 'discount rate'
 - Create a 'complete LCC data' dataframe for use in UC2
 - Drop all but 'year' and 'NPV' columns
 - Result is 'LCC cashflow' dataframe
- B) F2 - Aggregation function
- Build empty dataframe
 - Group asset register columns by 'asset_id'
 - Loop through group to assign variables to columns headers (inputs to F1)
 - Call the F1 - LCC analysis function
 - Concatenate the results into a 'cashflow' dataframe
 - Sum on 'year' column as new 'aggregated cashflow' dataframe
 - Result is the 'aggregated cashflow' dataframe
- C) Call the F2 - Aggregation function
- D) Plot the 'aggregated cashflow' dataframe
- E) Sum 'years' of 'aggregated cashflow' for final 'TCO NPV' value and print
- F) Data Exploration
- Plot the 'cashflow' dataframe to review anomalies etc. (non-aggregated)
 - Filter the 'complete LCC data' dataframe as required (on 'risk', 'condition' etc.)

Function 2 – Compare options

- A) (Option A) Life Cycle Cost Analysis for asset selected from asset register
- Select asset data based on specifying an 'asset_id'
 - Call the F1 - LCC analysis function
 - Plot the 'aggregated cashflow' dataframe as a line chart
 - Plot the phases breakdown
 - Using the 'complete LCC data' sum the columns based on life cycle phases taxonomy (CapEx, OpEx, EoL) into new 'phase breakdown' dataframe
 - Plot the 'phase breakdown' dataframe as a pie chart
 - Print a final 'LCC NPV' value by summing 'aggregated cashflow'
- B) (Option B) Life Cycle Cost Analysis for alternative strategy selected from Altern asset data
- Select asset data based on specifying an 'asset_id'
 - Call the F1 - LCC analysis function
 - Prepare the phases breakdown data for Option B
 - Using the 'complete LCC data' sum the columns based on life cycle phases taxonomy (CapEx, OpEx, EoL) into new 'phase breakdown' dataframe
 - Prepare a final 'LCC NPV' value by summing 'aggregated cashflow'
- C) Comparative analysis
- Plot the 'LCC cashflow' of Option A and Option B as a single line chart
 - Plot the 'phase breakdown' for option A
 - Plot the 'phase breakdown' for option B
 - Print the final 'LCC NPV' values for options A and B

Function 3 – Performance gap

- A) Run steps A, B and C from Use Case 1 on the Asset register
- B) Run steps A, B and C from Use Case 1 on the Asset register (Baseline)
- C) Plot the 'aggregated cashflow' dataframes from previous steps on a single line chart
- D) Provide final NPV TCO figures and variance
- Sum 'years' of 'aggregated cashflow' for final 'TCO NPV' value for each version of the asset register and print
 - Compare difference between version of the asset register and print

Data sources

- User inputs
- Asset register
- Alternative asset data
- Asset register (Baseline)

The prototype functions were developed in Python 3 using the Pandas and Matplotlib packages. Used in conjunction with the reference architecture, the plain language coding steps can facilitate software development in other programming languages.