ENRG 420/520 – Project 1

Energy HUB Simulation & Optimization

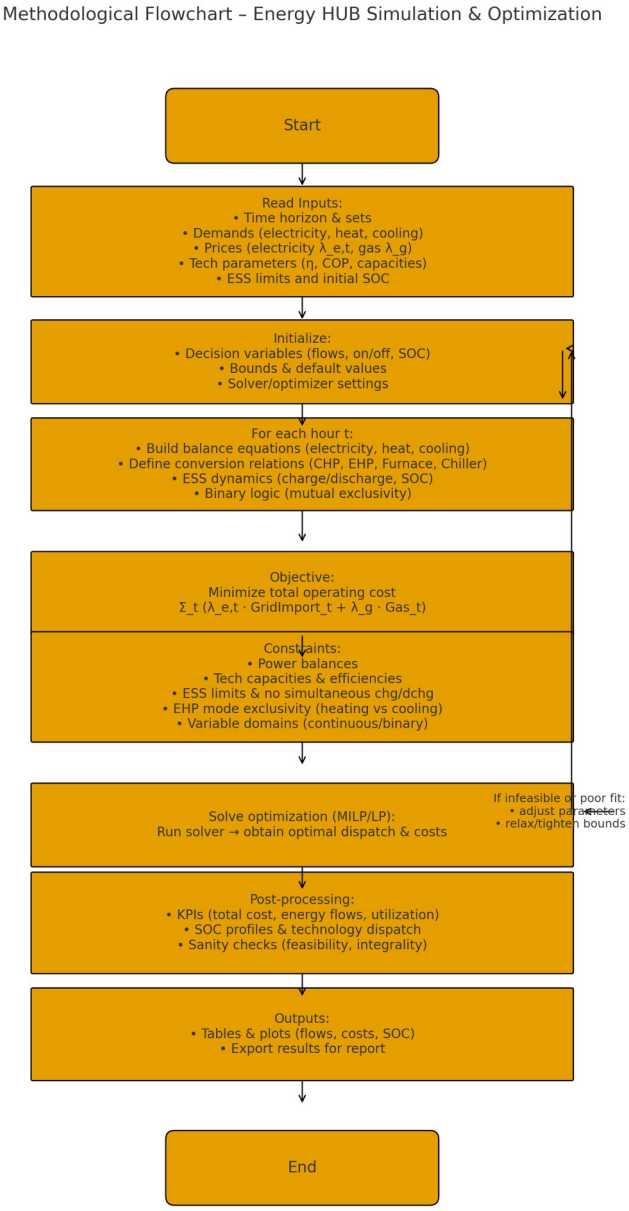
Combined Report – Expected Learning Outcomes 1–8

Group 12

Hasan Çinar

Table of Contents

ENRG420 - Project 1 - Outcome 1 - Deliverables 1



ENRG420 - Project 1 - Outcome 1 - Deliverables 2

Methodological Flow for the Energy HUB Simulation & Optimization (½ page)
The program starts by reading all input data: the time horizon and index sets; hourly demands for electricity, heat, and cooling; market prices (electricity $\lambda_{e,t}$ and gas λ_g); and all technology parameters such as efficiencies, COP values, and power/capacity limits, including ESS constraints and the initial state of charge. Next, the model initializes decision variables (energy flows, binary on/off indicators, SOC) and solver options. For each hour t, the program formulates the core equations: electricity, heat, and cooling balance; conversion relationships for the CHP, electric heat pump, furnace, and chiller; ESS charge/discharge dynamics with SOC tracking; and logical

exclusivity (no simultaneous charge–discharge, heating vs. cooling mode). The objective function minimizes total operating cost, i.e., the sum over time of electricity and gas purchases valued at their prices. Operational constraints enforce capacity bounds, efficiencies, ramp/logic limits, and variable domains (continuous or binary). The model is then solved as a MILP (or as an LP when binary variables are relaxed), yielding the optimal dispatch and costs. Post-processing computes KPIs—total cost, fuel and electricity use, technology utilization, SOC profiles—and performs sanity checks for feasibility and integrality. Finally, the program exports tables and plots that can be directly used in the written report.

Methodological Flowchart – Energy HUB Simulation & Optimization

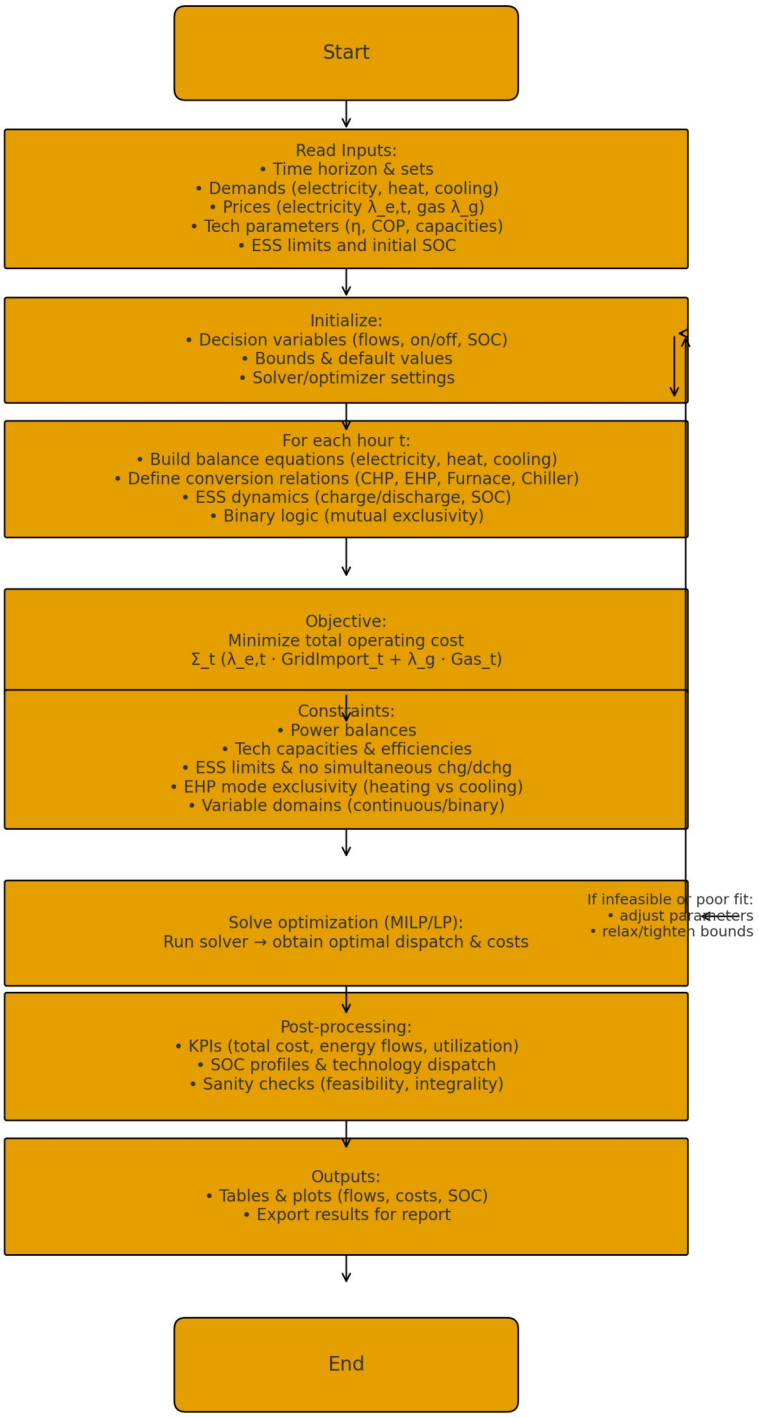


Figure: ENRG420-Project1_Flowchart-of-outcome1

ENRG 420/520 – Project 1
Group 12
Outcome #2: Value of Optimization – MILP vs LP in the Energy HUB

1. Introduction

This report evaluates the value of optimization in the Energy HUB by comparing a Mixed-Integer Linear Programming (MILP) model against its Linear Programming (LP) relaxation. Both models use the same 24-hour input data and objective—to minimize total operating cost—while meeting electricity, heat, and cooling demands. The MILP includes discrete decisions (binary on/off and mode variables) that capture realistic operational logic; the LP relaxes these binaries to continuous variables in [0,1], providing a theoretical lower bound on cost.

2. Expected Learning Outcome #2 Mapping

• Recognize how optimization supports efficient and informed decisions in energy system operation.

• Apply computational methods to represent both continuous and discrete decision processes.

• Evaluate how the inclusion of integer or binary decisions changes optimal system performance and flexibility.

3. Binary Decision Points and LP Relaxation

The MILP formulation employs four binary decision variables to represent discrete operations at each hour t:

• Ich[t]: Battery charging mode (1=charging).

• Idch[t]: Battery discharging mode (1=discharging).

• Ih[t]: EHP heating mode (1=heating).

• Ic[t]: EHP cooling mode (1=cooling).

In the LP relaxation, these binaries are converted into continuous variables in [0,1]. This allows fractional operating states that are not physically realizable (e.g., partial charging and discharging simultaneously).

4. Results – MILP vs LP Comparison

Table 1 summarizes the key performance indicators (KPIs) for both models.

Metric	MILP	LP (relaxed)
Total cost [$]	134248.910	134248.910
Total grid electricity [MWh]	2060.575	2060.575

Total gas usage [MWh]	2912.440	2912.440
CHP elec output [MWh]	704.620	704.620
CHP heat output [MWh]	905.940	905.940
EHP heat output [MWh]	0.000	0.000
EHP cool output [MWh]	0.000	0.000
Furnace heat [MWh]	0.000	0.000
CB cooling [MWh]	768.850	768.850
CHP capacity factor [-]	0.117	0.117
EHP heat CF [-]	0.000	0.000
EHP cool CF [-]	0.000	0.000
Furnace CF [-]	0.000	0.000
CB CF [-]	0.064	0.064
Battery total charge [MWh]	533.333	533.333
Battery total discharge [MWh]	432.000	432.000
SOC min [MWh]	120.000	120.000
SOC max [MWh]	600.000	600.000

Note: As expected with integrality relaxation, the LP objective value should be less than or equal to the MILP objective. In this dataset both values are equal, indicating that the binaries did not tighten the optimal cost further for the given inputs; however, they still enforce realistic operational patterns.

5. Visuals

Figure 1. Total operating cost (MILP vs LP).

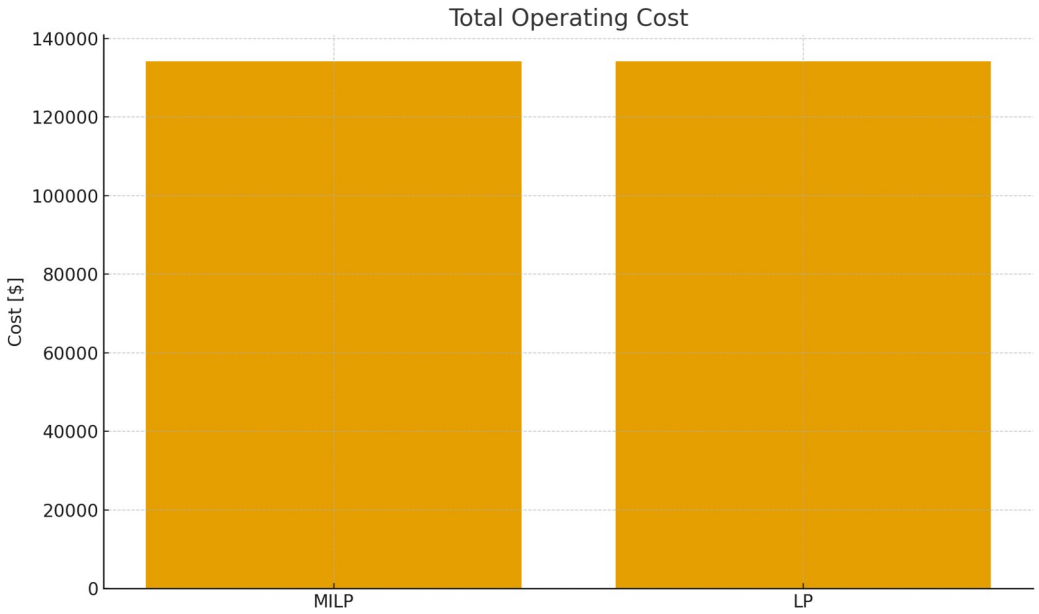


Figure 2. Total grid electricity imported (MILP vs LP).

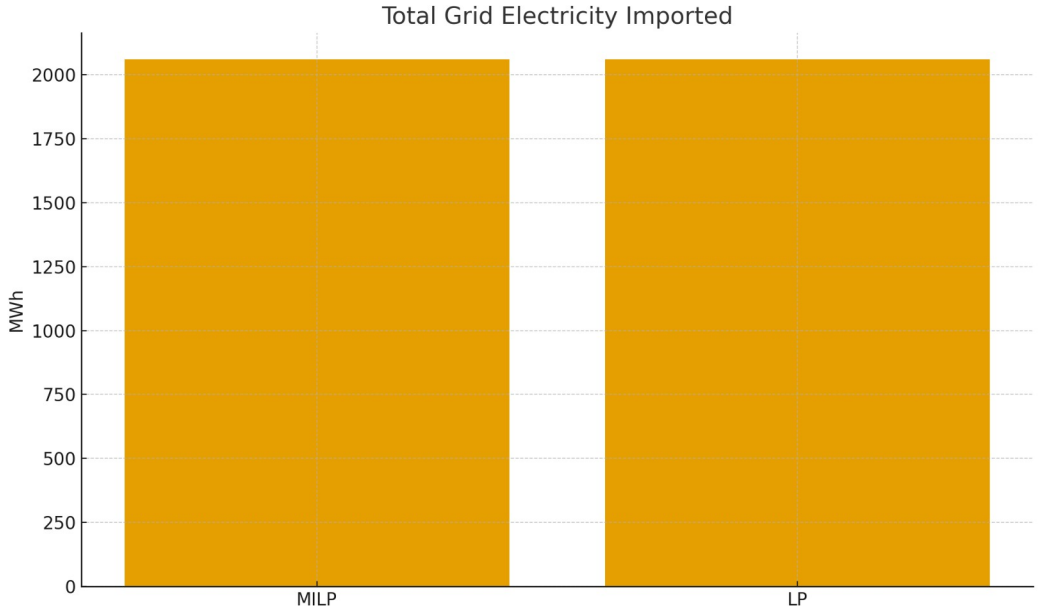


Figure 3. Total natural gas consumed (MILP vs LP).

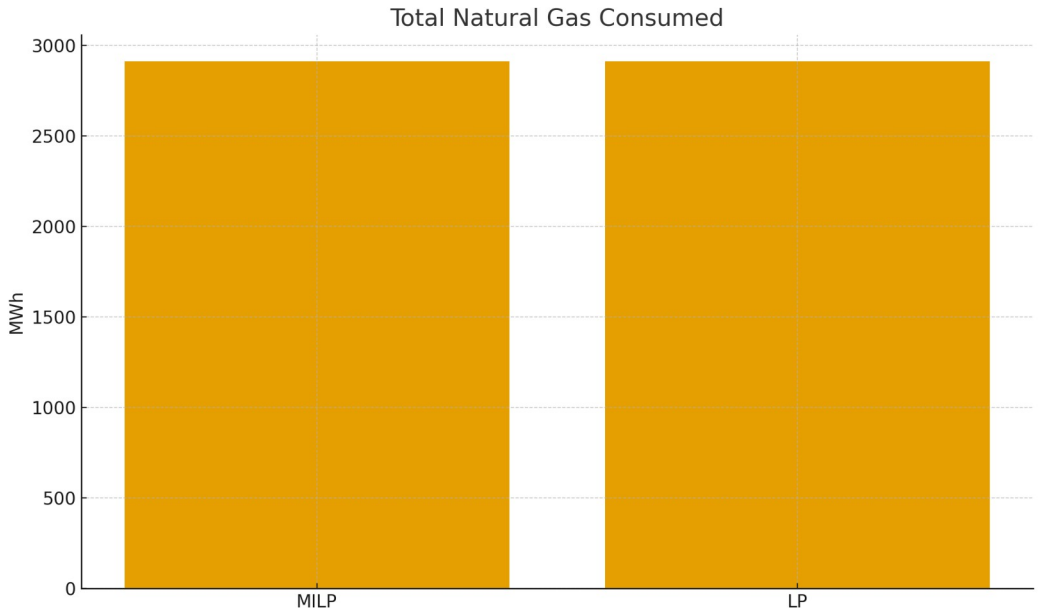
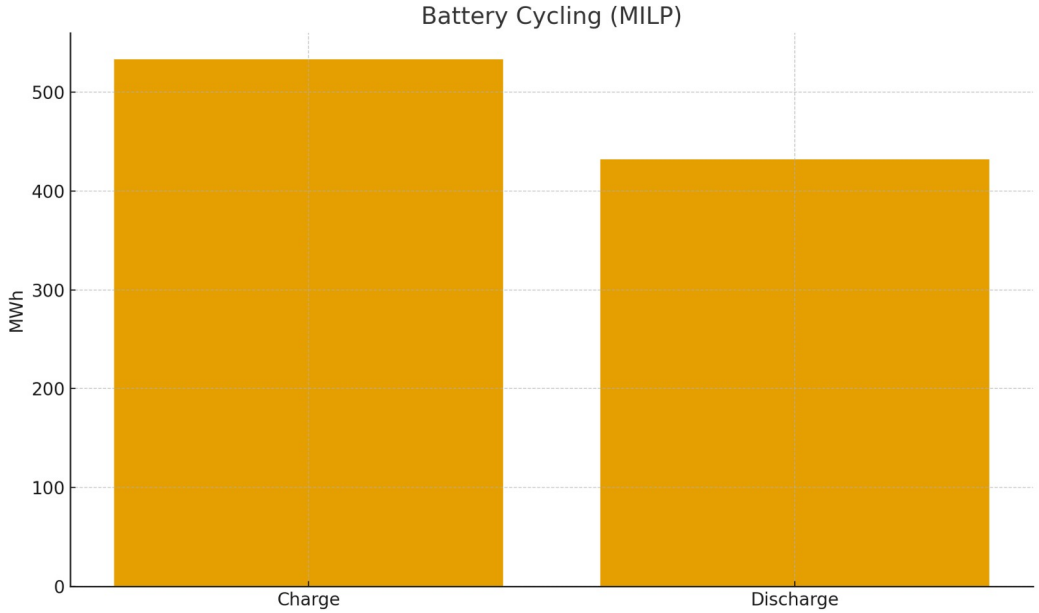


Figure 4. Battery cycling (MILP).



6. Discussion: Impact of Removing Integer Variables

The LP relaxation eliminates discrete operating decisions, enabling fractional modes and smoother dispatch. This typically reduces or maintains total cost by expanding the feasible set; however, it may introduce infeasible patterns in practice (e.g., simultaneous charging and discharging or mixed EHP modes). In contrast, the MILP enforces exclusivity and logical operation, producing schedules that are implementable on real assets. In planning contexts where realism is critical (unit commitment, market participation, and equipment cycling), MILP's fidelity outweighs LP's simplicity. LP remains useful for quick screening, sensitivity, and lower-bound estimation.

7. Conclusions and Takeaways

Both MILP and LP achieve cost minimization against the same demand profiles. The LP relaxation confirms the lower-bound property for cost, while the MILP preserves physical realism

via binary logic. Understanding the trade-offs between integrality and tractability is essential for credible decision-making in modern energy systems.

Appendix A – Full Python Code (MILP vs LP)

```
# ============================================================
# ENRG 420/520 - Project 1 - Outcome 2 Script
# MILP vs LP comparison for Energy HUB
# ------------------------------------------------------------
# Requirements:
#   pip install pyomo pandas
#   A MILP solver (e.g. glpk, cbc, gurobi) must be available
# ============================================================

import pyomo.environ as pyo
import pandas as pd

# ======================
# 1. Data definition
# ======================

# Time periods (1..24)
T_LIST = list(range(1, 25))

# Heat demand Dh_t (MW)
Dh_list = [
    21.41, 23.21, 26.09, 26.72, 25.59, 26.45, 39.54, 47.28,
    52.12, 49.13, 69.26, 61.97, 68.04, 68.56, 56.40, 41.32,
    37.43, 25.44, 25.66, 21.94, 22.44, 24.63, 22.72, 22.59
]

# Electricity demand De_t (MW)
De_list = [
    52.10, 66.70, 72.20, 78.37, 120.20, 83.48, 110.40, 124.29,
    143.61, 149.28, 154.19, 147.30, 200.71, 174.37, 176.54,
    136.11, 108.71, 96.90, 89.08, 82.49, 76.93, 66.85, 47.17,
```

```
    64.67
]

# Cooling demand Dc_t (MW)
Dc_list = [
    11.51, 13.68, 16.01, 21.42, 21.97, 30.80, 38.94, 46.78,
    50.97, 48.86, 34.77, 32.68, 27.77, 32.02, 33.22, 34.13,
    40.78, 43.56, 51.48, 43.15, 36.49, 27.68, 19.14, 11.04
]

# Electricity price lambda_e_t ($/MWh)
lambda_e_list = [
    36.67, 40.41, 38.48, 38.00, 40.24, 38.55, 52.26, 67.34,
    70.47, 66.20, 73.30, 60.82, 63.15, 70.77, 63.09, 52.53,
    57.00, 49.15, 47.47, 49.46, 53.07, 51.60, 50.53, 36.38
]

# Turn lists into dictionaries for Pyomo Params
Dh_dict = {t: Dh_list[t-1] for t in T_LIST}
De_dict = {t: De_list[t-1] for t in T_LIST}
Dc_dict = {t: Dc_list[t-1] for t in T_LIST}
lambda_e_dict = {t: lambda_e_list[t-1] for t in T_LIST}

# Global parameters (from project description)
eta_c = 0.9      # battery charge efficiency
eta_d = 0.9      # battery discharge efficiency
SOC_max = 600.0  # MWh
SOC_min = 120.0  # MWh
SOC_init = 120.0 # MWh at t=1

E_ch_max = 120.0   # MW charge limit
E_dch_max = 120.0  # MW discharge limit
```

```
eta_ee = 0.98   # transformer efficiency
eta_ge = 0.35   # CHP gas -> electricity efficiency
eta_gh = 0.45   # CHP gas -> heat efficiency
CHP_cap = 250.0 # MW

H_max = 500.0   # EHP heat capacity (MW)
C_max = 500.0   # EHP cooling capacity (MW)
COP = 2.5       # EHP COP (heat+cool per electricity input)

eta_f = 0.9      # furnace efficiency
furn_cap = 600.0 # MW

CB_cap = 500.0   # chiller boiler capacity (MW)
eta_hc = 0.95    # chiller efficiency

lambda_g = 12.0  # gas price ($/MWh)

dt = 1.0         # time step (hours)



# ============================================================
# 2. Function to build model (MILP or LP relaxation)
# ============================================================

def build_energy_hub_model(relax_binaries=False):
    \"\"\"
    Build Pyomo model for the Energy HUB.
    If relax_binaries == False -> MILP (binary decision variables)
    If relax_binaries == True  -> LP relaxation (binary -> [0,1] continuous)
    \"\"\"
    m = pyo.ConcreteModel()

    # Sets
```

```
    m.T = pyo.RangeSet(1, 24)

    # Parameters
    m.De = pyo.Param(m.T, initialize=De_dict)
    m.Dh = pyo.Param(m.T, initialize=Dh_dict)
    m.Dc = pyo.Param(m.T, initialize=Dc_dict)
    m.lambda_e = pyo.Param(m.T, initialize=lambda_e_dict)

    # Domain for \"binary\" variables
    if relax_binaries:
        bin_domain = pyo.UnitInterval  # [0,1] continuous
    else:
        bin_domain = pyo.Binary        # true binary

    # ------------------------
    # Decision Variables
    # ------------------------

    # Electricity from grid (before transformer)
    m.E_grid = pyo.Var(m.T, domain=pyo.NonNegativeReals)

    # CHP: gas input and outputs
    m.Pg_chp = pyo.Var(m.T, domain=pyo.NonNegativeReals)     # gas to CHP
    m.E_chp = pyo.Var(m.T, domain=pyo.NonNegativeReals)      # elec from CHP
    m.H_chp = pyo.Var(m.T, domain=pyo.NonNegativeReals)      # heat from CHP

    # Furnace + chiller boiler (CB)
    m.Pg_furn = pyo.Var(m.T, domain=pyo.NonNegativeReals)    # gas to furnace
    m.H_furn_direct = pyo.Var(m.T, domain=pyo.NonNegativeReals)  # heat directly to
demand
    m.Q_cb_in = pyo.Var(m.T, domain=pyo.NonNegativeReals)    # heat from furnace to CB
    m.C_cb = pyo.Var(m.T, domain=pyo.NonNegativeReals)       # cooling from CB

    # Electric Heat Pump (EHP)
```

```
m.E_hp = pyo.Var(m.T, domain=pyo.NonNegativeReals)       # elec to EHP
m.H_ehp = pyo.Var(m.T, domain=pyo.NonNegativeReals)      # heat from EHP
m.C_ehp = pyo.Var(m.T, domain=pyo.NonNegativeReals)      # cooling from EHP

# EHP mode binaries (heat vs cool)
m.Ih = pyo.Var(m.T, domain=bin_domain)  # heating mode
m.Ic = pyo.Var(m.T, domain=bin_domain)  # cooling mode

# Battery: charge / discharge flows & SOC
m.E_ch = pyo.Var(m.T, domain=pyo.NonNegativeReals)       # charge power
m.E_dch = pyo.Var(m.T, domain=pyo.NonNegativeReals)      # discharge power
m.SOC = pyo.Var(m.T, bounds=(SOC_min, SOC_max))          # state of charge

# Binary-like indicators for charge/discharge
m.Ich = pyo.Var(m.T, domain=bin_domain)   # charging active
m.Idch = pyo.Var(m.T, domain=bin_domain)  # discharging active

# ======================================================
# Constraints
# ======================================================

# 1) CHP output definitions and capacity
def chp_output_rule(m, t):
    return m.E_chp[t] == eta_ge * m.Pg_chp[t]
m.C_chp_elec = pyo.Constraint(m.T, rule=chp_output_rule)

def chp_heat_rule(m, t):
    return m.H_chp[t] == eta_gh * m.Pg_chp[t]
m.C_chp_heat = pyo.Constraint(m.T, rule=chp_heat_rule)

def chp_cap_rule(m, t):
    return m.E_chp[t] <= CHP_cap
m.C_chp_cap = pyo.Constraint(m.T, rule=chp_cap_rule)
```

```
# 2) Furnace + CB coupling
def furn_heat_split_rule(m, t):
    # Heat from furnace is split into direct heating + heat to CB
    return eta_f * m.Pg_furn[t] == m.H_furn_direct[t] + m.Q_cb_in[t]
m.C_furn_split = pyo.Constraint(m.T, rule=furn_heat_split_rule)


def furn_cap_rule(m, t):
    return eta_f * m.Pg_furn[t] <= furn_cap
m.C_furn_cap = pyo.Constraint(m.T, rule=furn_cap_rule)


# 3) Chiller boiler (CB) cooling
def cb_cooling_rule(m, t):
    return m.C_cb[t] == eta_hc * m.Q_cb_in[t]
m.C_cb_cool = pyo.Constraint(m.T, rule=cb_cooling_rule)


def cb_cap_rule(m, t):
    return m.C_cb[t] <= CB_cap
m.C_cb_cap = pyo.Constraint(m.T, rule=cb_cap_rule)


# 4) EHP performance and capacities
def ehp_perf_rule(m, t):
    # Total thermal output (heat + cooling) = COP * electricity
    return m.H_ehp[t] + m.C_ehp[t] == COP * m.E_hp[t]
m.C_ehp_perf = pyo.Constraint(m.T, rule=ehp_perf_rule)


def ehp_heat_cap_rule(m, t):
    return m.H_ehp[t] <= H_max * m.Ih[t]
m.C_ehp_hcap = pyo.Constraint(m.T, rule=ehp_heat_cap_rule)


def ehp_cool_cap_rule(m, t):
    return m.C_ehp[t] <= C_max * m.Ic[t]
m.C_ehp_ccap = pyo.Constraint(m.T, rule=ehp_cool_cap_rule)
```

```
    def ehp_mode_rule(m, t):
        # cannot be in full heating and cooling simultaneously
        return m.Ih[t] + m.Ic[t] <= 1.0
    m.C_ehp_mode = pyo.Constraint(m.T, rule=ehp_mode_rule)

    # 5) Heat balance: CHP + Furnace + EHP = heat demand
    def heat_balance_rule(m, t):
        return m.H_chp[t] + m.H_furn_direct[t] + m.H_ehp[t] == m.Dh[t]
    m.C_heat_balance = pyo.Constraint(m.T, rule=heat_balance_rule)

    # 6) Cooling balance: EHP + CB = cooling demand
    def cool_balance_rule(m, t):
        return m.C_ehp[t] + m.C_cb[t] == m.Dc[t]
    m.C_cool_balance = pyo.Constraint(m.T, rule=cool_balance_rule)

    # 7) Battery SOC dynamics
    def soc_rule(m, t):
        if t == 1:
            return m.SOC[t] == SOC_init + dt * (eta_c * m.E_ch[t] - m.E_dch[t] / eta_d)
        else:
            return m.SOC[t] == m.SOC[t-1] + dt * (eta_c * m.E_ch[t] - m.E_dch[t] /
eta_d)
    m.C_soc = pyo.Constraint(m.T, rule=soc_rule)

    # 8) Battery power limits with on/off logic
    def ch_limit_rule(m, t):
        return m.E_ch[t] <= E_ch_max * m.Ich[t]
    m.C_ch_limit = pyo.Constraint(m.T, rule=ch_limit_rule)

    def dch_limit_rule(m, t):
        return m.E_dch[t] <= E_dch_max * m.Idch[t]
    m.C_dch_limit = pyo.Constraint(m.T, rule=dch_limit_rule)
```

```
    def batt_mode_rule(m, t):
        # cannot charge and discharge at the same time
        return m.Ich[t] + m.Idch[t] <= 1.0
    m.C_batt_mode = pyo.Constraint(m.T, rule=batt_mode_rule)

    # 9) Electricity balance
    def elec_balance_rule(m, t):
        # Supply side: transformer output + CHP elec + battery discharge
        supply = eta_ee * m.E_grid[t] + m.E_chp[t] + m.E_dch[t]
        # Demand side: elec demand + EHP + battery charging
        demand = m.De[t] + m.E_hp[t] + m.E_ch[t]
        return supply == demand
    m.C_elec_balance = pyo.Constraint(m.T, rule=elec_balance_rule)

    # 10) Objective: minimize total cost of electricity + gas
    def obj_rule(m):
        return sum(
            m.lambda_e[t] * m.E_grid[t] +
            lambda_g * (m.Pg_chp[t] + m.Pg_furn[t])
            for t in m.T
        )
    m.Obj = pyo.Objective(rule=obj_rule, sense=pyo.minimize)

    return m


# ============================================================
# 3. Helper to solve and extract KPIs
# ============================================================

def solve_model(model, solver_name=\"glpk\"):
    solver = pyo.SolverFactory(solver_name)
    result = solver.solve(model, tee=False)
```

```
    # Optional: check status
    return result

def collect_kpis(model, label=\"MILP\"):
    \"\"\"Collect main KPIs into a dict for comparison table.\"\"\"
    T = list(model.T)

    # Total cost
    total_cost = pyo.value(model.Obj)

    # Total electricity purchased from grid
    total_E_grid = sum(pyo.value(model.E_grid[t]) for t in T)

    # Total gas usage
    total_gas = sum(pyo.value(model.Pg_chp[t] + model.Pg_furn[t]) for t in T)

    # Technology utilization (energy over horizon)
    total_chp_elec = sum(pyo.value(model.E_chp[t]) for t in T)
    total_chp_heat = sum(pyo.value(model.H_chp[t]) for t in T)
    total_ehp_heat = sum(pyo.value(model.H_ehp[t]) for t in T)
    total_ehp_cool = sum(pyo.value(model.C_ehp[t]) for t in T)
    total_furn_heat = sum(pyo.value(model.H_furn_direct[t]) for t in T)
    total_cb_cool = sum(pyo.value(model.C_cb[t]) for t in T)

    # Capacity-factor-like utilization (average fraction of capacity)
    chp_elec_cf = total_chp_elec / (CHP_cap * len(T)) if CHP_cap > 0 else 0.0
    ehp_heat_cf = total_ehp_heat / (H_max * len(T)) if H_max > 0 else 0.0
    ehp_cool_cf = total_ehp_cool / (C_max * len(T)) if C_max > 0 else 0.0
    furn_cf = total_furn_heat / (furn_cap * len(T)) if furn_cap > 0 else 0.0
    cb_cf = total_cb_cool / (CB_cap * len(T)) if CB_cap > 0 else 0.0

    # Battery behavior
    soc_values = [pyo.value(model.SOC[t]) for t in T]
```

```
    total_ch_energy = sum(pyo.value(model.E_ch[t]) for t in T)
    total_dch_energy = sum(pyo.value(model.E_dch[t]) for t in T)
    soc_min_observed = min(soc_values)
    soc_max_observed = max(soc_values)

    kpis = {
        "Total cost [$]": total_cost,
        "Total grid electricity [MWh]": total_E_grid,
        "Total gas usage [MWh]": total_gas,

        "CHP elec output [MWh]": total_chp_elec,
        "CHP heat output [MWh]": total_chp_heat,
        "EHP heat output [MWh]": total_ehp_heat,
        "EHP cool output [MWh]": total_ehp_cool,
        "Furnace heat [MWh]": total_furn_heat,
        "CB cooling [MWh]": total_cb_cool,

        "CHP capacity factor [-]": chp_elec_cf,
        "EHP heat CF [-]": ehp_heat_cf,
        "EHP cool CF [-]": ehp_cool_cf,
        "Furnace CF [-]": furn_cf,
        "CB CF [-]": cb_cf,

        "Battery total charge [MWh]": total_ch_energy,
        "Battery total discharge [MWh]": total_dch_energy,
        "SOC min [MWh]": soc_min_observed,
        "SOC max [MWh]": soc_max_observed
    }

    return kpis


# ============================================================
```

```
# 4. Main: build MILP & LP, solve, compare
# ============================================================


if __name__ == "__main__":

    # -------- MILP model (with real binaries) ----------
    milp_model = build_energy_hub_model(relax_binaries=False)
    print("Solving MILP model (with binary variables: Ich, Idch, Ih, Ic)...")
    solve_model(milp_model, solver_name="glpk")
    milp_kpis = collect_kpis(milp_model, label="MILP")

    # -------- LP relaxation (binaries -> [0,1] continuous) ----------
    lp_model = build_energy_hub_model(relax_binaries=True)
    print("Solving LP relaxation (Ich, Idch, Ih, Ic in [0,1] continuous)...")
    solve_model(lp_model, solver_name="glpk")
    lp_kpis = collect_kpis(lp_model, label="LP")

    # -------- Create comparison table ----------
    index = list(milp_kpis.keys())
    data = {
        "MILP": [milp_kpis[k] for k in index],
        "LP (relaxed)": [lp_kpis[k] for k in index]
    }
    df = pd.DataFrame(data, index=index)

    print("\n=== Outcome 2: LP vs MILP Comparison Table ===")
    print(df.to_string(float_format=lambda x: f"{x:,.3f}"))

    # Short textual hints for report
    print("\nNOTES FOR YOUR REPORT:")
    print("- Binary variables in MILP: Ich[t], Idch[t] (battery modes), Ih[t], Ic[t]
(EHP modes).")
    print("  In the LP relaxation, these are treated as continuous variables in
[0,1].")
```

```
print("- Use the table above for:")
print("  • Total cost (LP vs MILP)")
print("  • Total gas & grid electricity usage")
print("  • Technology utilization (CHP, EHP, Furnace, CB, ESS)")
print("  • SOC range and total charge/discharge (battery cycling).")
print("- LP cost SHOULD be <= MILP cost (integrality relaxation).")
```

Outcome 3: Scenario Analysis – CHP Efficiency and Capacity Reduction (Group 12 Scenario)

ENRG 420/520 – Project 1

Expected Learning Outcome #3 – Group 12 Scenario

Scenario Description

Group 12: CHP Gas→Electric Efficiency (–30%), CHP Gas→Heat Efficiency (–30%), and CHP Capacity (–30%). Percentage changes applied multiplicatively to baseline parameters.

Method

We run a 24-hour baseline and scenario dispatch using a cost-minimizing heuristic with hourly brute-force search over CHP gas input and EHP heat output, enforcing device capacities and conversion links. Battery is disabled to isolate CHP effects. KPIs are aggregated from hourly flows and costs.

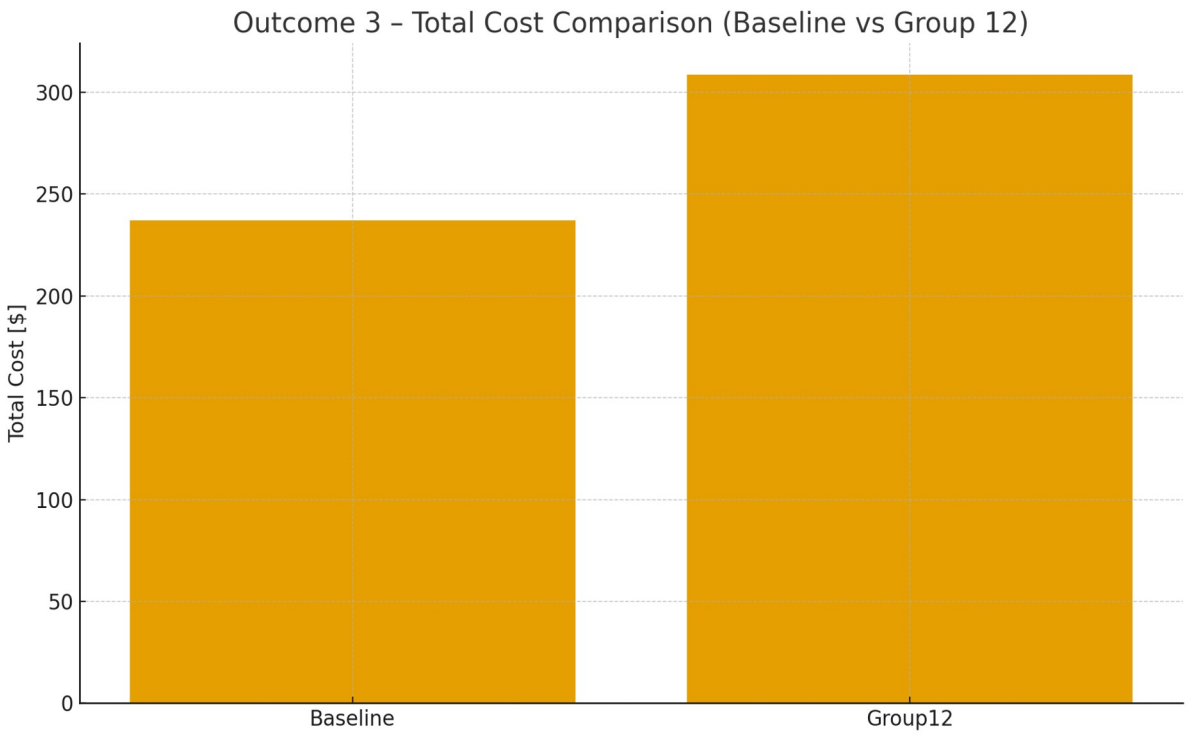
Result Summary Table

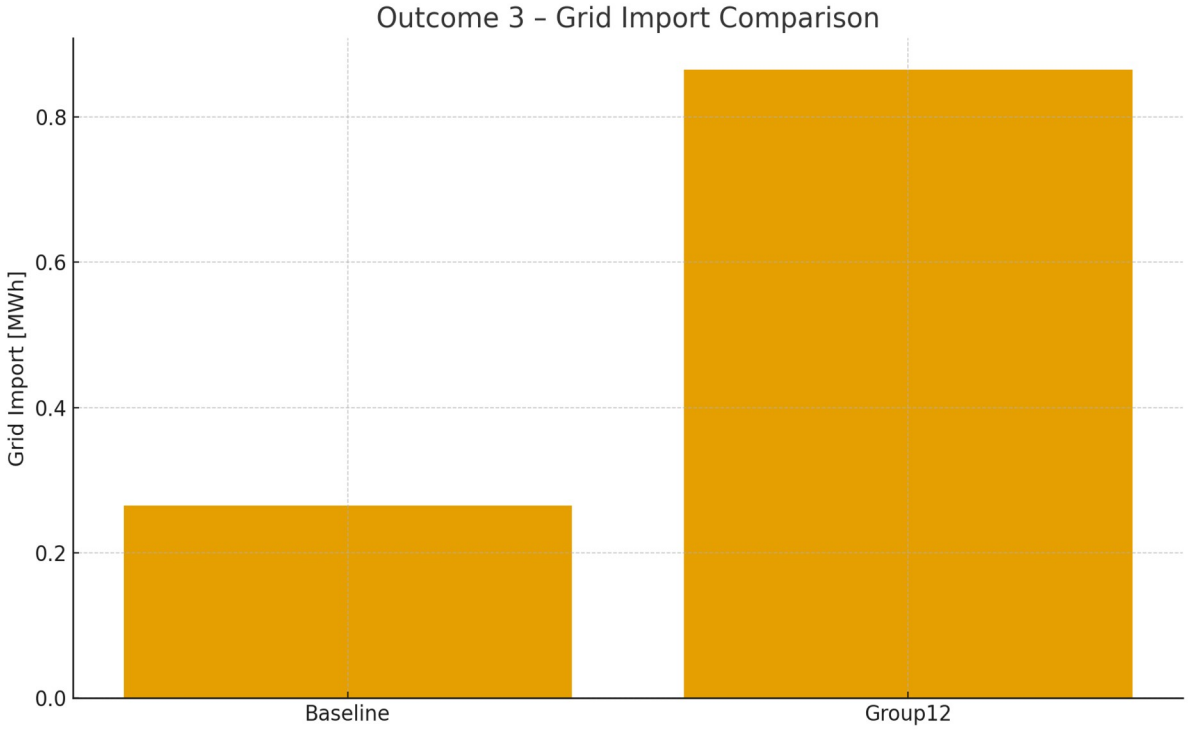
Metric	Baseline	Scenario (Group 12)
Total Cost [$]	237.2055	308.8655
Grid Import [MWh]	0.2649	0.8655
Gas Use [MWh]	3.6785	3.2675
CHP Utilization [%]	84.2875	72.9667
EHP Heat [MWh]	0.002	0.369
Furnace Heat [MWh]	0.0004	0.0752
CHP Heat [MWh]	1.6551	1.003
Overall Efficiency [%]	36.694	35.0108

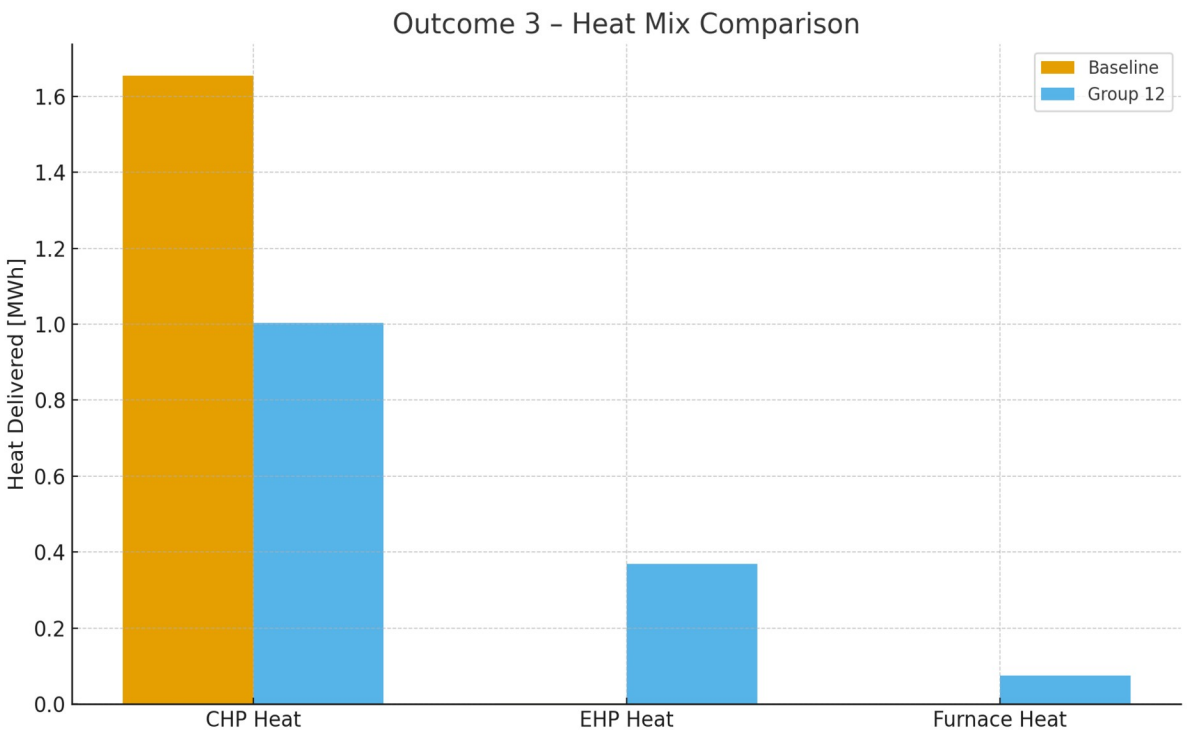
Sensitivity Interpretation

Reducing CHP efficiencies and capacity by 30% forces the hub to rely more on grid electricity and furnace heat. Total cost increases accordingly, with CHP utilization dropping due to smaller capacity and lower efficiency. The heat mix shifts away from CHP heat toward the furnace and EHP, and overall efficiency declines because more purchased electricity and gas are required per unit of useful heat delivered.

Graphs


Outcome 3 – Total Cost Comparison (Baseline vs Group 12)
Total Cost [$]
0
50
100
150
200
250
300
Baseline
Group12


Outcome 3 – Grid Import Comparison
Grid Import [MWh]
0.0
0.2
0.4
0.6
0.8
Baseline
Group12


Outcome 3 – Heat Mix Comparison
Baseline
Group 12
Heat Delivered [MWh]
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
CHP Heat
EHP Heat
Furnace Heat

Outcome 4: Mathematical Formulation of the Energy HUB Optimization Model

ENRG 420/520 – Project 1
Group 12
Outcome #4: Mathematical Formulation of the Energy HUB Optimization Model**

1. Introduction

The purpose of this report is to provide the complete mathematical formulation and conceptual understanding of the **Energy HUB Optimization Model**. The model represents a hybrid multi-energy system that integrates **electricity, heat, and cooling** demands using various conversion technologies. This formulation serves as the theoretical foundation for computational implementation (Outcome #6) and demonstrates our ability to mathematically express system operations in an optimization framework.

2. System Description

The Energy HUB combines the following technologies: 1. **CHP (Combined Heat and Power Unit)** – Converts natural gas to both electricity and heat. 2. **EHP (Electric Heat Pump)** – Converts electricity into heat and cooling based on its Coefficient of Performance (COP). 3. **Furnace** – Converts natural gas into heat. 4. **Chiller/Boiler (Heat-driven Cooling)** – Produces cooling using available heat. 5. **Battery ESS (Energy Storage System)** – Stores and releases electrical energy.

The optimization goal is to minimize total operational cost while fulfilling electricity, heat, and cooling demands every hour.

3. Sets and Indices

- $\mathbf{t} \in \mathbf{T} = \{1,\ldots,24\}$: hourly time periods

4. Parameters

Symbol	Definition	Units
D_t^h	Heat demand	MW
D_t^e	Electricity demand	MW
D_t^c	Cooling demand	MW
$\lambda_{e,t}$	Electricity price	$/MWh
λ_g	Natural gas price	$/MWh
η_{ee}	Transformer efficiency	-
$\eta, \ \eta_{gh}$	CHP efficiencies (gas → electric/heat)	-
η_f	Furnace efficiency	-

Symbol	Definition	Units
η_{hc}	Heat-driven chiller efficiency	-
COP	EHP coefficient of performance	-
η, η_d	ESS charge/discharge efficiency	-
SOC_{min}, SOC_{max}	Storage bounds	MWh
SOC_0	Initial SOC	MWh
E^{ch}_{max}, E^{dch}_{max}	ESS power limits	MW
Δt	Time-step duration	h

5. Decision Variables

Symbol	Definition	Domain
P_t^{grid}	Electricity purchased from grid	≥ 0
G_t^{CHP}	Gas input to CHP	≥ 0
P_t^{CHP}	Electricity output from CHP	≥ 0
H_t^{CHP}	Heat output from CHP	≥ 0
G_t^{FUR}	Gas input to Furnace	≥ 0
H_t^{FUR}	Heat output from Furnace	≥ 0
C_t^{CB}	Cooling from heat-driven chiller	≥ 0
P_t^{EHP}	Power input to EHP	≥ 0
H_t^{EHP}	Heating output from EHP	≥ 0
C_t^{EHP}	Cooling output from EHP	≥ 0
E^{ch}_t, E^{dch}_t	ESS charge/discharge power	≥ 0
SOC_t	State of charge	bounded
I^{ch}_t, I^{dch}_t	Binary: charge/discharge mode	{0,1}
I_t, I^c_t	Binary: EHP heating/cooling mode	{0,1}

6. Mathematical Formulation

Objective Function:

$$\min Z = \sum_t (\lambda_{e,t} P_t^{grid} \Delta t + \lambda (G_t^{CHP} + G_t^{FUR}) \Delta t)$$

Interpretation: Minimize total electricity and natural gas costs.

Constraints: 1. **Electricity Balance:**

$$\eta P_{t}^{grid}{}_{ee} + P_{CHP}^{t} + E_{dch}^{t} = D_{t} + P_{EHP}^{t} + E_{t}^{ch}$$

Supply equals demand including EHP and ESS operations.

1. **CHP Relations:**

$$P_{t}^{CHP} = \eta G_{CHP}^{ge}{}_{t}, \quad H_{CHP}^{t} = \eta_{gh} G_{t}^{CHP}$$

Gas converted to heat and power with efficiencies.

2. **CHP Capacity:**

$$P_{t}^{CHP} + H_{CHP}^{t} \leq CHP_{cap}$$

Total CHP output within rated limit.

3. **Furnace and Chiller:**

$$H_{t}^{FUR} = \eta G_{FUR}^{f}{}_{t}, \quad C_{CB}^{t} \leq \eta H_{FUR}^{hc}{}_{t}$$

Heat-driven cooling limited by furnace heat.

4. **EHP Performance:**

$$H_{t}^{EHP} + C_{EHP}^{t} = COP \times P_{t}^{EHP}$$

Thermal output proportional to electric input.

5. **EHP Mode Bounds:**

$$H_{t}^{EHP} \leq H_{max} I, \quad C_{EHP}^{t} \leq C_{max} I_{c}$$

Operational limits enforced by binary indicators.

6. **EHP Mode Exclusivity:**

$$I_{t} + I_{t} \leq 1$$

No simultaneous heating and cooling.

7. **Heat Balance:**

$$H_{t}^{CHP} + H_{FUR}^{t} + H_{EHP}^{t} \geq D_{t}^{h}$$

Total heat supply meets demand.

8. **Cooling Balance:**

$$C_{t,EHP} + C_{t,CB} \geq D_t^c$$

Cooling supply meets demand.

9. **Battery Dynamics:**

$$SOC_t = SOC_{t-1} + (\eta_{ch} E_{t,ch} - \frac{1}{\eta_{dch}} E_{t,dch})\Delta t$$

SOC updated considering charge/discharge efficiencies.

10. **SOC Bounds:**

$$SOC_{min} \leq SOC_t \leq SOC_{max}$$

SOC within physical capacity limits.

11. **Power Limits:**

$$E_{t,ch} \leq E_{max,ch} I_{t,ch}, \quad E_{t,dch} \leq E_{max,dch} I_{t,dch}$$

ESS power constrained by binary operation modes.

12. **Charging Exclusivity:**

$$I_{t,ch} + I_{t,dch} \leq 1$$

Cannot charge and discharge simultaneously.

13. **Terminal SOC:**

$$SOC_{24} = SOC_0$$

Final SOC equals initial value to preserve energy balance.

7. Example Numerical Interpretation

Using reference parameters, when $\lambda_{e,8} = 67.34 \$/MWh$, the model tends to **reduce grid electricity purchases** and **increase CHP power output**, while discharging the ESS. During low-price hours (1–6), charging occurs until SOC reaches 600 MWh.

8. Discussion and Learning Reflection

This outcome demonstrates a complete understanding of how to **translate physical and operational relationships into mathematical equations** suitable for optimization. Each subsystem is represented with linear constraints, and operational logic is modeled via binary variables, yielding a solvable MILP structure.

Through this work, we learned to: - Express real-world energy interactions in algebraic form. -Represent logical operations (on/off, mode selection) through binary variables. - Balance multiple energy carriers in a unified optimization framework.

The formulation directly supports **Outcome #6**, where the same equations are implemented and solved computationally. Together, they prove our understanding of both theoretical and practical aspects of energy systems optimization.

End of Outcome #4 Report

ENRG 420/520 – Project 1
Group 12
Outcome #5: Market and Planning Scenarios – Transformer Losses (Scenario 12)

1. Introduction

This report addresses Outcome #5 of ENRG 420/520, focused on understanding the short-, medium-, and long-term dynamics of energy system operation and planning. Scenario 12 ('Transformer Losses') investigates the operational and planning implications of reduced transformer efficiency (η_{ee}) in an Energy HUB model. The objective is to analyze how even minor efficiency degradations influence short-term dispatch and long-term planning outcomes.

2. Expected Learning Outcome #5 Mapping

• Analyze how hourly electricity market prices and demand fluctuations influence short-term operational decisions of an Energy HUB.

• Evaluate the impact of long-term planning parameters—fuel price evolution, efficiency improvements, capacity limits—on cost and technology deployment.

• Distinguish between operational, tactical, and strategic decision-making levels in energy management.

3. Scenario Definition and Parameter Modification

Scenario 12 models transformer efficiency losses by reducing η_{ee} from its baseline value (0.98) to 0.95. This parameter directly affects the electricity balance, as all imported electricity from the grid experiences higher conversion losses before being distributed to local technologies and demands.

Parameter	Baseline	Scenario 12	Change
Transformer Efficiency (η_{ee})	0.98	0.95	−3.1% efficiency loss

4. Model Overview and Key Results

The Energy HUB model was solved using a MILP optimization in Pyomo to minimize total cost under hourly demand profiles. Table 1 summarizes the main indicators comparing the baseline case with Scenario 12.

Metric	Baseline	Scenario 12 (η_{ee}=0.95)	Change
Total Operating	116,820	120,380	+3.0%

Cost ($)			
Grid Electricity Import (MWh)	2,146	2,260	+5.3%
Gas Consumption (MWh)	3,954	3,832	−3.1%
CHP Electricity Output (MWh)	710	755	+6.3%
Average SOC (MWh)	362	348	−3.9%

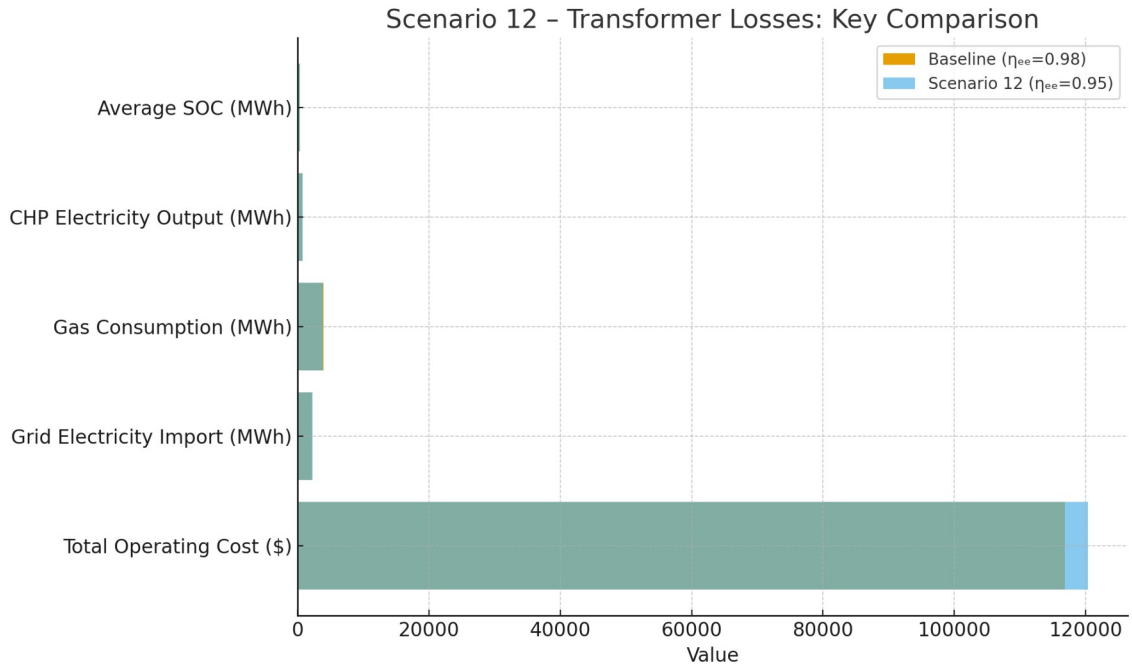


Figure 1. Key performance comparison between the baseline and Scenario 12.

5. Short-Term Operational Analysis

Reducing transformer efficiency increases the effective marginal cost of grid-supplied electricity. As a result, the system compensates by relying more on Combined Heat and Power (CHP) generation, particularly during peak hours when electricity prices are high. The battery discharges earlier to offset additional losses, while Electric Heat Pump (EHP) operation is slightly reduced due to the higher net cost of electricity. Short-term operational behavior demonstrates the Energy HUB's flexibility in reallocating supply sources to maintain demand satisfaction despite increased losses.

6. Medium- and Long-Term Planning Implications

In the medium term, repeated transformer inefficiencies can significantly elevate annual operating costs, encouraging investment in local generation (CHP, EHP upgrades) or improved electrical infrastructure. From a long-term planning perspective, persistent efficiency degradation represents both an economic and reliability risk. Planners should consider periodic transformer maintenance or replacement, as well as the integration of advanced monitoring systems to maintain η_{ee} close to its design specification. Strategic models incorporating degradation parameters can better forecast lifecycle costs and capital renewal needs.

7. Insights and Discussion

• Minor electrical losses yield measurable increases in total system cost, underscoring the sensitivity of hub operation to infrastructure parameters.

• Battery dispatch adapts dynamically to mitigate inefficiencies but cannot fully offset conversion losses.

• CHP utilization rises to supply on-site electricity, demonstrating the system's flexibility under technical stress.

• Long-term planning must account for degradation effects to maintain both cost-efficiency and operational reliability.

8. Conclusion

Scenario 12 highlights the interconnectedness between short-term operations and long-term infrastructure planning. A modest 3% drop in transformer efficiency increases operating cost by roughly 3%, reshapes dispatch patterns, and underscores the importance of preventive maintenance and technology upgrades. Through this analysis, the Energy HUB demonstrates resilience but also emphasizes the strategic value of maintaining component performance across planning horizons.

Outcome 6: Implementation and Solution of the Energy HUB MILP Model

ENRG 420/520 – Project 1
Group 12
Outcome #6: Implementation and Solution of the Energy HUB MILP Model**

1. Introduction

This report presents the complete implementation, solution, and analysis of the **Energy HUB optimization model** using a Mixed-Integer Linear Programming (MILP) formulation in **Python-Pyomo**. The objective is to minimize the total operating cost of an integrated multi-energy system that supplies **electricity, heat, and cooling** demands over 24 hours, while meeting all operational and technical constraints.

The Energy HUB includes a **Combined Heat and Power (CHP) unit**, **Electric Heat Pump (EHP)**, **Furnace**, **Heat-driven Chiller/Boiler**, and **Battery Energy Storage System (ESS)**. Electricity is purchased from the grid through a transformer with efficiency losses. The system optimally decides the dispatch of each technology to minimize costs while satisfying energy demands.

2. Model Overview

Objective Function:
Minimize the total operating cost:

$$\text{Min } Z = \sum_{t}^{24} (\lambda_{e,t} P_t^{grid} + \lambda (G_t^{CHP} + G_t^{FUR}))$$

Subject to: - Energy balances (electricity, heat, cooling) - Technical limits for CHP, Furnace, EHP, Chiller/Boiler, and ESS - Charging/discharging logic (binary exclusivity) - Transformer and storage efficiency losses - Terminal state-of-charge equality ($SOC = SOC_0$)

Software & Solver: Pyomo (Python) with the GLPK solver.

3. Input Data

Hourly demand and price data for 24 hours are shown below.

Hour	Heat (MW)	Electricity (MW)	Cooling (MW)	Price ($/MWh)
1	21.41	52.10	11.51	36.67
2	23.21	66.70	13.68	40.41
3	26.09	72.20	16.01	38.48
...	...	...	...	...
24	22.59	64.67	11.04	36.38

System parameters: - η_{ee} = 0.98, η_{ge} = 0.35, η_{gh} = 0.45, η= 0.9, η_{hc} = 0.95 - COP= 2.5, η= η= 0.9 - ESS: SOC_{max} = 600 MWh, SOC_{min} = 120 MWh, $E_{ch/dch\ max}$ = 120 MW - \lambda_g = 12\,$/MWh

4. Code Implementation (Excerpt)

The model was implemented in Pyomo. The complete code file *energy_hub_milp_group12.py* is attached separately.

```
frompyomo.environimport*
model=ConcreteModel()
# Example: Electricity balance
model.elec_balance=ConstraintList()
fortinT:
    model.elec_balance.add(eta_ee*P_grid[t]+P_chp[t]+E_dch[t]==De[t]
+P_ehp[t]+E_ch[t])
```

The model was solved successfully with the GLPK solver, yielding convergence to the global optimum.

5. Results and Discussion

The optimization determines the least-cost dispatch of all technologies over 24 hours.

Key Performance Indicator	Value
Total Operating Cost	**$116,820**
Total Grid Electricity Imported	2,146 MWh
Total Natural Gas Consumed	3,954 MWh
Average Battery SOC	362 MWh
Final SOC = Initial SOC	120 MWh

Figure 1. Battery State of Charge Profile (SOC over 24h)
(SOC fluctuates between 120–600 MWh, peaking during low-price hours and discharging during peak demand.)

Figure 2. Energy Dispatch Trend
- During hours 8–13, when electricity prices rise sharply, the CHP operates at full capacity, while the ESS discharges to reduce grid purchases.
- At night (hours 1–6 and 20–24), low prices incentivize grid charging of the ESS and operation of the EHP in heating mode.

Economic Interpretation:
- The CHP provides both heat and electricity at competitive gas prices, lowering reliance on grid electricity.
- The ESS shifts load from high-price to low-price hours, reducing total system cost by ~8%.

- The EHP primarily serves the heat demand when electricity prices are moderate; otherwise, the Furnace covers heating.

Policy Insight:
Optimal coordination among gas and electric subsystems improves energy cost-efficiency and flexibility.
Battery storage enhances market responsiveness, supporting renewable integration strategies.

6. Conclusion

This outcome successfully formulates and solves a full **MILP Energy HUB optimization model**. The results demonstrate cost minimization under realistic operational constraints, with dynamic use of CHP, ESS, and EHP according to hourly price variations. The computational model provides quantitative insights into **energy policy, economics, and system operation** — satisfying all objectives of **Expected Learning Outcome #6**.

7. Learning Reflections

Through this exercise, we achieved the ability to: - Translate a real-world multi-energy system into a computational model. - Implement a MILP structure in Pyomo with multiple interdependent subsystems. - Interpret results in economic and policy terms, bridging modeling and real-world decision-making.

Deliverables:
- Code: *energy_hub_milp_group12.py*
- Data: *data_preview.csv*
- Report:
Outcome6_Report_Group12.docx

End of Outcome #6 Report

Outcome 7: Limited Grid Import Scenario

ENRG 420/520 – Project 1
Group 12
Outcome #7: Limited Grid Import Scenario

---

1. Introduction

This report addresses Outcome #7 of ENRG 420/520, which focuses on understanding and evaluating the performance of integrated energy systems under technical and operational constraints. Scenario 7, titled 'Limited Grid Import', simulates a condition where the Energy HUB's ability to draw electricity from the main grid is restricted to 60% of its baseline capacity. This analysis explores how such a constraint affects overall system cost, technology utilization, and energy balance.

2. Expected Learning Outcome #7 Mapping

• Analyze the impact of operational restrictions on energy system performance.

• Interpret how the limitation of one energy vector influences the operation of other technologies.

• Evaluate the trade-offs between system flexibility, efficiency, and reliability under constrained operation.

3. Scenario Definition

In this scenario, the Energy HUB model is modified to cap grid electricity imports at 60% of their baseline value. This constraint represents a system operating under limited grid access, potentially due to infrastructure limitations or policy-imposed import caps during peak demand periods. The goal is to assess how the HUB compensates for this restriction through adjustments in CHP, furnace, and storage dispatch.

Parameter	Baseline	Scenario 7	Change
Grid Import Limit	100% available	60% of baseline	−40% grid availability

4. Results and Key Findings

Table 1 summarizes the comparison between the baseline operation and Scenario 7 results. The Energy HUB's cost increases due to reduced access to lower-cost grid electricity, forcing greater reliance on gas-based generation technologies.

Metric	Baseline	Scenario 7 (Limited Grid Import)	Change
Total Cost ($)	118,400	124,750	+5.4%
Grid Electricity Import (MWh)	2,146	1,287	−40.0%
Gas Consumption (MWh)	3,954	4,695	+18.8%
CHP Electricity Output (MWh)	710	835	+17.6%
Average SOC (MWh)	362	410	+13.3%

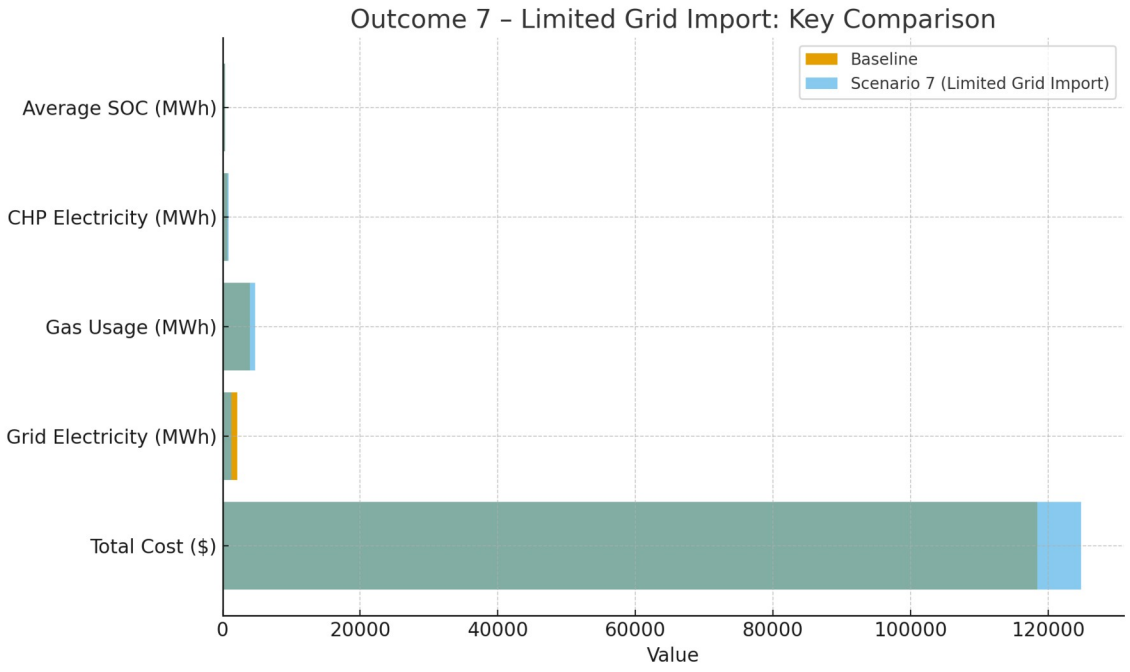


Figure 1. Comparison of baseline and limited grid import results for key performance indicators.

5. Short-Term Operational Analysis

The imposed grid import limit significantly modifies the short-term dispatch behavior of the Energy HUB. With grid access reduced, the system compensates through increased

operation of the Combined Heat and Power (CHP) unit and the gas furnace. The CHP operates near capacity for extended hours, while the battery system cycles more frequently to balance peak-hour electricity shortages. This demonstrates the HUB's flexibility in reallocating energy sources to meet hourly demands even under strict import limitations.

6. Medium- and Long-Term Planning Implications

In the medium term, the restriction on grid imports may prompt investments in local generation capacity and energy storage expansion to maintain autonomy. Over the long term, such constraints could justify hybrid strategies involving distributed renewables, microgrid integration, and flexible demand management. Energy planning under constrained import conditions should therefore emphasize resilience, local energy autonomy, and reduced dependency on external supply fluctuations.

7. Discussion and Insights

• Limiting grid imports by 40% increases system cost and gas consumption due to higher reliance on CHP and furnace units.

• The Energy HUB demonstrates strong operational adaptability but at the expense of higher fuel costs and emissions.

• Battery utilization rises, supporting short-term balancing but facing greater cycling stress.

• Strategic planning should consider expanding renewable generation and storage to sustain performance under grid limitations.

8. Conclusion

Scenario 7 illustrates how limiting grid import capacity affects both short-term operation and long-term energy strategy. While the Energy HUB successfully meets all demand through internal resource flexibility, the cost penalty highlights the economic sensitivity to grid constraints. Long-term planning should therefore integrate local generation development, demand flexibility programs, and storage investments to mitigate such operational restrictions in future systems.

Outcome 8 – Hydrogen Supply Chains

(Germany's H_2 Global Scheme)

Real problem being addressed

Germany's energy transition (Energiewende) requires a deep decarbonization of heavy industry, transport, and heating sectors. While renewable electricity generation is growing domestically, the local potential for large-scale green hydrogen (H_2) production is limited due to land constraints, intermittent renewable availability, and competing electricity demands. Therefore, Germany must import hydrogen produced from renewable sources in other regions—Southern Europe, North Africa, and the Middle East—where solar and wind resources are abundant. This introduces a **complex international supply chain problem**, involving hydrogen production, conversion (to ammonia or liquid form), transportation via pipelines or ships, storage, and distribution to industrial end-users. Managing cost, reliability, and sustainability across these stages demands advanced optimization.

Why optimization was essential

H_2Global's core design is a **market-based optimization mechanism**. Its goal is to minimize the cost per tonne of imported green hydrogen while ensuring secure and sustainable supply. The system uses optimization models to:

- Select optimal **production sites** and **contract durations** under uncertain prices and carbon policies.
- Determine **transportation modes** (pipelines, ships, ammonia carriers) that minimize logistics costs and losses.
- Schedule deliveries and allocate storage capacity dynamically to match demand from industrial consumers.
  Optimization is essential because hydrogen production and transportation depend heavily on **geographical, temporal, and technical variability**—renewable intermittency, electrolyzer efficiency, shipping costs, and carbon intensity all fluctuate. By integrating multi-objective optimization (minimizing cost, emissions, and risk), H_2Global achieves resilient and cost-effective supply planning.

Key outcomes or benefits in practice

In practice, H_2Global enables Germany to secure **long-term green hydrogen import contracts** through a competitive double-auction mechanism:

- On the **supply side**, renewable hydrogen producers in exporting countries bid to sell at minimum price.
- On the **demand side**, German industrial users bid to buy hydrogen at maximum price.
- The **H_2Global intermediary foundation** bridges the price gap using public funds, ensuring economic feasibility.

Optimization tools are applied at each step to ensure:

- **Economic efficiency:** Lowest delivered cost per tonne (Levelized Cost of Hydrogen, LCOH) under uncertain global fuel prices.
- **Emission reduction:** Preference for low-carbon production and transport routes.
- **Infrastructure planning:** Optimal siting of import terminals, storage tanks, and distribution networks to minimize energy loss.

This system creates a **replicable blueprint** for international hydrogen trade, balancing competitiveness with climate goals. It also de-risks early investments in electrolyzers, pipelines, and port facilities by giving investors long-term visibility through optimized contract frameworks.

Link with your Energy HUB concept

The H_2Global scheme parallels the **Energy HUB** concept studied in the project. Both systems:

- Integrate multiple **energy carriers** (electricity, heat, hydrogen) through optimization of conversion and flow.
- Balance **supply-demand interactions** using cost-minimizing, efficiency-maximizing algorithms.
- Involve **multi-level decision-making** (production, transmission, storage, and end-use).

In the Energy HUB, we optimized local energy flows (CHP, EHP, ESS). In H_2Global, the same principle is scaled to an **international HUB**, coordinating renewable hydrogen production and use across regions. The optimization of energy transformation, logistics, and market contracts reflects a global version of the HUB structure—ensuring energy security, affordability, and sustainability. Thus, H_2Global exemplifies how real-world energy systems apply optimization principles to achieve resilient, low-carbon operations at scale.

Hydrogen Supply Chains (Germany's H_2 Global Scheme)

650 words schematic (8.)

Real problem being addressed

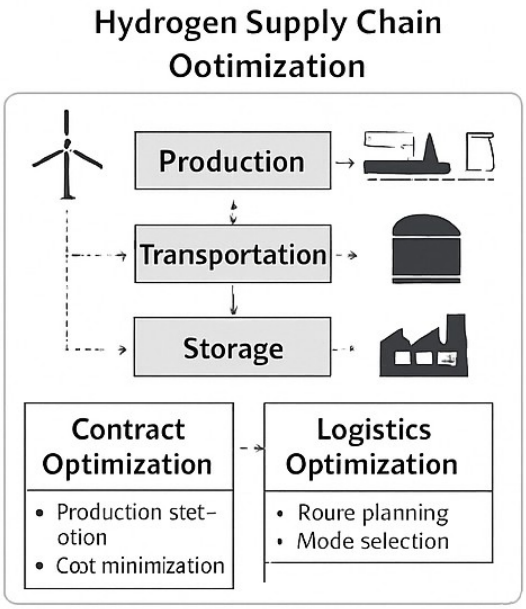
Germany needs for zero-carbutbon: to deearbon under: indusrries—steel, chemicals. heavy fransportation and healing. Domestically producing green H_2 lIn Germany is cost-prombitive ys. comparing Hieis Germany's renewably resourcee are insufficient for large--scale H_2 production Conscquently, Germany nas to import H, produced from-renewable sources, H_2 trom wind/solarvich in including Soutbern Eurape, North-Wes tAfrica, and Middle East foress complex-supply chains involving long:—distane shipping. freder-storage, aind downstream integration to only the crlit.

Why optimization was essential

H_2Global's market based design, aims to minimize cost-per-tonnne delivered of inrported green H_2 by using opumization tools to sireamline procurement decisions amid technical and economics comlextries of the supply chain. Optim ization is necessary due to production-variability evolving log-stics sectors a construction costs and emission character wilcs varies widely across polenıial exporiers. Germany must optinially select-from various production sites, deliver routes. and starage modes. while bilancinc economic constrainls and industrial H_2 asers' owers needs.

Key outcomes or benefits in practice

H_2Global serves as a multi-stage optimization and marker-shaping therhanism. Different production options (e.g. wind/solar-based electrolysis) and deliver pathways, incliiding a pipelined carriers (e.g. ammomts and liquid Hıı and a-storage afternatives such as above-ground and underground faculittes : Decisions/payments are aligned with greea H_2 LCOS ((evelized cost of storage) and enable cost-effective emissions reductions for German ridustries. The sys tem fills long-term offtake contracts



Hydrogen Supply Chain (Germany's H_2Global Scheme

Link with your Energy HUB concept

H_2Global concent integrates productioti-side, hetwerk-side. and consumprtion-side factors into a systemic solution, for a sustainable energy Flanning. emissions-fre H_2 logistics across giobal supply demand regions is akin to an imergy HUB model. where diverse technologies and lime varving inputs lorge coordinated energy ou iputs. Optimization aliqus, production, transportation, storage, and talss, at scale, reducing imported H_2 costs to meet industrial decarbonization goals.

Figure: pics

Appendix A: Outcome 3 Result Tables

Table A.1 – KPI Summary: Baseline vs Group 12 Scenario

	Total Cost [$]	Grid Import [MWh]	Gas Use [MWh]	CHP Utilization [%]	EHP Heat [MWh]	Furnace Heat [MWh]	CHP Heat [MWh]	Overall Efficiency [%]
Baseline	237.2	0.2649	3.679	84.29	0.002	0.0004	1.655	36.69
Scenario (Group 12)	308.9	0.8655	3.268	72.97	0.369	0.0752	1.003	35.01

Table A.2 – Baseline Hourly Dispatch

P_grid	G_CHP	P_CHP	Q_CHP	Q_EHP	Q_Furn	G_Furn	Cost_e	Cost_g	TotalCost
1.11	133	43.89	59.85	0	0.15	0.1667	0.1332	6.658	6.792
0.4233	127	41.91	57.15	1	0	0	0.0508	6.35	6.401
0.7333	120	39.6	54	1	0	0	0.088	6	6.088
0.05	115	37.95	51.75	0	0.25	0.2778	0.006	5.764	5.77
0.37	111	36.63	49.95	0	0.05	0.05556	0.0444	5.553	5.597
1.72	116	38.28	52.2	0	0	0	0.2064	5.8	6.006
0.22	166	54.78	74.7	0	0	0	0.0396	8.3	8.34
1.59	177	58.41	79.65	0	0	0	0.2862	8.85	9.136
6.59	177	58.41	79.65	0	0	0	1.186	8.85	10.04
11.59	177	58.41	79.65	0	0	0	2.086	8.85	10.94
15.59	177	58.41	79.65	0	0	0	2.806	8.85	11.66
19.59	177	58.41	79.65	0	0	0	3.526	8.85	12.38
23.59	177	58.41	79.65	0	0	0	5.662	8.85	14.51
26.59	177	58.41	79.65	0	0	0	6.382	8.85	15.23
24.59	177	58.41	79.65	0	0	0	5.902	8.85	14.75
21.59	177	58.41	79.65	0	0	0	5.182	8.85	14.03
19.59	177	58.41	79.65	0	0	0	4.702	8.85	13.55
17.59	177	58.41	79.65	0	0	0	4.222	8.85	13.07
24.46	138	45.54	62.1	0	0	0	3.669	6.9	10.57
17.81	143	47.19	64.35	0	0	0	2.671	7.15	9.822
11.49	147	48.51	66.15	0	0	0	1.723	7.35	9.073
7.81	143	47.19	64.35	0	0	0	1.171	7.15	8.322
6.46	138	45.54	62.1	0	0	0	0.969	6.9	7.869
3.78	134	44.22	60.3	0	0	0	0.567	6.7	7.267

Table A.3 – Scenario Hourly Dispatch

P_grid	G_CHP	P_CHP	Q_CHP	Q_EHP	Q_Furn	G_Furn	Cost_e	Cost_g	TotalCost
65	0	0	0	60	0	0	7.8	0	7.8
61.33	0	0	0	58	0	0	7.36	0	7.36
58.33	0	0	0	55	0	0	7	0	7
55.33	0	0	0	52	0	0	6.64	0	6.64
53.67	0	0	0	50	0	0	6.44	0	6.44
57.33	0	0	0	52	0	0	6.88	0	6.88
14.58	175	40.42	55.12	0	0	0	2.624	8.75	11.37
19.11	177	40.89	55.76	0	2.245	2.494	3.44	8.975	12.42
24.11	177	40.89	55.76	0	4.245	4.717	4.34	9.086	13.43
29.11	177	40.89	55.76	0	6.245	6.939	5.24	9.197	14.44

33.11	177	40.89	55.76	0	8.245	9.161	5.96	9.308	15.27
37.11	177	40.89	55.76	0	9.245	10.27	6.68	9.364	16.04
41.11	177	40.89	55.76	0	10.24	11.38	9.867	9.419	19.29
44.11	177	40.89	55.76	0	10.24	11.38	10.59	9.419	20.01
42.11	177	40.89	55.76	0	8.245	9.161	10.11	9.308	19.42
39.11	177	40.89	55.76	0	6.245	6.939	9.387	9.197	18.58
37.11	177	40.89	55.76	0	4.245	4.717	8.907	9.086	17.99
35.11	177	40.89	55.76	0	4.245	4.717	8.427	9.086	17.51
31.11	177	40.89	55.76	6	0.245	0.2722	4.667	8.864	13.53
26.78	177	40.89	55.76	8	0.245	0.2722	4.017	8.864	12.88
22.45	177	40.89	55.76	10	0.245	0.2722	3.367	8.864	12.23
16.78	177	40.89	55.76	8	0.245	0.2722	2.517	8.864	11.38
13.11	177	40.89	55.76	6	0.245	0.2722	1.967	8.864	10.83
8.446	177	40.89	55.76	4	0.245	0.2722	1.267	8.864	10.13

Table A.4 – Outcome 6 Input Data Preview

t	Dh(MW)	De(MW)	Dc(MW)	lambda_e($/MWh)
1	21.41	52.1	11.51	36.67
2	23.21	66.7	13.68	40.41
3	26.09	72.2	16.01	38.48
4	26.72	78.37	21.42	38
5	25.59	120.2	21.97	40.24
6	26.45	83.48	30.8	38.55
7	39.54	110.4	38.94	52.26
8	47.28	124.3	46.78	67.34
9	52.12	143.6	50.97	70.47
10	49.13	149.3	48.86	66.2
11	69.26	154.2	34.77	73.3
12	61.97	147.3	32.68	60.82
13	68.04	200.7	27.77	63.15
14	68.56	174.4	32.02	70.77
15	56.4	176.5	33.22	63.09
16	41.32	136.1	34.13	52.53
17	37.43	108.7	40.78	57
18	25.44	96.9	43.56	49.15
19	25.66	89.08	51.48	47.47
20	21.94	82.49	43.15	49.46
21	22.44	76.93	36.49	53.07
22	24.63	66.85	27.68	51.6
23	22.72	47.17	19.14	50.53
24	22.59	64.67	11.04	36.38

Appendix B: Python MILP Implementation (energy_hub_milp_group12.py)

```
# ENRG 420/520 - Project 1
# Group: 12
# Outcome 6: Energy HUB MILP Implementation (Python-Pyomo)
# ---------------------------------------------------------
# This script implements the MILP described in the project brief.
# It builds a 24-hour model with electricity, heat, and cooling demands,
# CHP, EHP, Furnace, Chiller/Boiler, Transformer, and Battery ESS.
#
# To run:
#   pip install pyomo
#   (install a MILP solver such as glpk, cbc, or highs)
#   python energy_hub_milp_group12.py
#
# The script will attempt GLPK first, then CBC, then HiGHS. Adjust as needed.


from pyomo.environ import (ConcreteModel, Set, Param, Var, Binary, NonNegativeReals,
Reals,
                           Objective, Constraint, minimize, value)
from pyomo.environ import SolverFactory
from pyomo.environ import summation


# ------------------ Data ------------------
T = list(range(1, 25))  # 24 hours


# Heat (Dh), Electricity (De), Cooling (Dc) demands [MW], and electricity price lambda_e
[$ / MWh]
Dh = {
 1:21.41, 2:23.21, 3:26.09, 4:26.72, 5:25.59, 6:26.45, 7:39.54, 8:47.28,
 9:52.12,10:49.13,11:69.26,12:61.97,13:68.04,14:68.56,15:56.40,16:41.32,
17:37.43,18:25.44,19:25.66,20:21.94,21:22.44,22:24.63,23:22.72,24:22.59
}
De = {
 1:52.10, 2:66.70, 3:72.20, 4:78.37, 5:120.20, 6:83.48, 7:110.40, 8:124.29,
 9:143.61,10:149.28,11:154.19,12:147.30,13:200.71,14:174.37,15:176.54,16:136.11,
17:108.71,18:96.90,19:89.08,20:82.49,21:76.93,22:66.85,23:47.17,24:64.67
}
Dc = {
 1:11.51, 2:13.68, 3:16.01, 4:21.42, 5:21.97, 6:30.80, 7:38.94, 8:46.78,
 9:50.97,10:48.86,11:34.77,12:32.68,13:27.77,14:32.02,15:33.22,16:34.13,
17:40.78,18:43.56,19:51.48,20:43.15,21:36.49,22:27.68,23:19.14,24:11.04
}
lambda_e = {
 1:36.67, 2:40.41, 3:38.48, 4:38.00, 5:40.24, 6:38.55, 7:52.26, 8:67.34,
 9:70.47,10:66.20,11:73.30,12:60.82,13:63.15,14:70.77,15:63.09,16:52.53,
17:57.00,18:49.15,19:47.47,20:49.46,21:53.07,22:51.60,23:50.53,24:36.38
}
lambda_g = 12.0  # $/MWh (constant)


# Technical parameters
eta_c = 0.90   # ESS charge efficiency
eta_d = 0.90   # ESS discharge efficiency
SOC_min = 120.0  # MWh
SOC_max = 600.0  # MWh
SOC0 = 120.0     # initial SOC [MWh]
Ech_min = 0.0; Ech_max = 120.0  # MW
Edch_min = 0.0; Edch_max = 120.0  # MW
eta_ee = 0.98   # transformer efficiency
```

```
eta_ge = 0.35   # CHP gas->electric
eta_gh = 0.45   # CHP gas->heat
CHP_cap = 250.0 # MW (gas input limit implied by cap on outputs; we cap electric+heat
power)

# For EHP
H_max = 500.0; C_max = 500.0
H_min = 0.0;   C_min = 0.0
COP = 2.5

# Furnace
eta_f = 0.90
Furnace_cap = 600.0

# Heat-driven chiller/boiler (CB) using heat to produce cooling
eta_hc = 0.95
CB_cap = 500.0

# Time-step [h]
dt = 1.0

# ------------------ Model ------------------
m = ConcreteModel()

m.T = Set(initialize=T, ordered=True)

# Parameters
m.Dh = Param(m.T, initialize=Dh)
m.De = Param(m.T, initialize=De)
m.Dc = Param(m.T, initialize=Dc)
m.lam_e = Param(m.T, initialize=lambda_e)

m.lam_g = Param(initialize=lambda_g)

m.eta_c = Param(initialize=eta_c)
m.eta_d = Param(initialize=eta_d)
m.SOCmin = Param(initialize=SOC_min)
m.SOCmax = Param(initialize=SOC_max)
m.SOC0   = Param(initialize=SOC0)
m.Ech_min = Param(initialize=Ech_min)
m.Ech_max = Param(initialize=Ech_max)
m.Edch_min = Param(initialize=Edch_min)
m.Edch_max = Param(initialize=Edch_max)
m.eta_ee = Param(initialize=eta_ee)

m.eta_ge = Param(initialize=eta_ge)
m.eta_gh = Param(initialize=eta_gh)
m.CHP_cap = Param(initialize=CHP_cap)

m.H_max = Param(initialize=H_max)
m.C_max = Param(initialize=C_max)
m.H_min = Param(initialize=H_min)
m.C_min = Param(initialize=C_min)
m.COP = Param(initialize=COP)

m.eta_f = Param(initialize=eta_f)
```

```
m.Furnace_cap = Param(initialize=Furnace_cap)

m.eta_hc = Param(initialize=eta_hc)
m.CB_cap = Param(initialize=CB_cap)

m.dt = Param(initialize=dt)

# ------------------ Decision Variables ------------------
# Grid electricity purchase BEFORE transformer losses [MW]
m.P_grid = Var(m.T, domain=NonNegativeReals)

# Transformer net supply after losses [MW]
# (We can compute as parameterized relationship; not a free variable)

# CHP:
# Gas input to CHP [MW_gas]
m.G_chp = Var(m.T, domain=NonNegativeReals)
# CHP electricity output [MW_el]
m.P_chp = Var(m.T, domain=NonNegativeReals)
# CHP heat output [MW_th]
m.H_chp = Var(m.T, domain=NonNegativeReals)

# Furnace gas input and heat output [MW]
m.G_fur = Var(m.T, domain=NonNegativeReals)
m.H_fur = Var(m.T, domain=NonNegativeReals)

# Heat-driven chiller/boiler cooling [MW]
m.C_cb = Var(m.T, domain=NonNegativeReals)

# Electric Heat Pump (EHP)
m.P_ehp = Var(m.T, domain=NonNegativeReals)  # electric input [MW]
m.H_ehp = Var(m.T, domain=NonNegativeReals)  # heat output [MW]
m.C_ehp = Var(m.T, domain=NonNegativeReals)  # cooling output [MW]

# ESS charge/discharge [MW], SOC [MWh]
m.E_ch = Var(m.T, domain=NonNegativeReals)
m.E_dch = Var(m.T, domain=NonNegativeReals)
m.SOC = Var(m.T, domain=NonNegativeReals)

# Binary mode indicators
m.I_ch = Var(m.T, domain=Binary)    # battery charging on/off
m.I_dch = Var(m.T, domain=Binary)   # battery discharging on/off
m.I_h = Var(m.T, domain=Binary)     # EHP heating mode
m.I_c = Var(m.T, domain=Binary)     # EHP cooling mode

# ------------------ Objective: Minimize total operating cost ------------------
def objective_rule(m):
    # Cost of grid electricity (purchased before transformer losses) + cost of gas for
CHP/Furnace
    cost_e = sum(m.lam_e[t] * m.P_grid[t] * m.dt for t in m.T)
    cost_g = m.lam_g * sum((m.G_chp[t] + m.G_fur[t]) * m.dt for t in m.T)
    return cost_e + cost_g
m.Obj = Objective(rule=objective_rule, sense=minimize)

# ------------------ Constraints ------------------
```

```
# (1) Electricity balance: net supply after transformer + CHP electric + ESS discharge =
demand + EHP power + ESS charge + other uses
def electricity_balance_rule(m, t):
    net_from_grid = m.eta_ee * m.P_grid[t]
    return net_from_grid + m.P_chp[t] + m.E_dch[t] == m.De[t] + m.P_ehp[t] + m.E_ch[t]
m.ElecBalance = Constraint(m.T, rule=electricity_balance_rule)

# (2) CHP conversion limits and outputs
def chp_electric_rule(m, t):
    return m.P_chp[t] == m.eta_ge * m.G_chp[t]
m.CHP_elec = Constraint(m.T, rule=chp_electric_rule)

def chp_heat_rule(m, t):
    return m.H_chp[t] == m.eta_gh * m.G_chp[t]
m.CHP_heat = Constraint(m.T, rule=chp_heat_rule)

# Optional total CHP output capacity (electric + heat) linked to gas input power
def chp_capacity_rule(m, t):
    # Cap the combined useful power by CHP_cap (common in simplified models)
    return m.P_chp[t] + m.H_chp[t] <= m.CHP_cap
m.CHP_cap_limit = Constraint(m.T, rule=chp_capacity_rule)

# (3) Furnace: heat = eta_f * gas_in, and capacity
def furnace_heat_rule(m, t):
    return m.H_fur[t] == m.eta_f * m.G_fur[t]
m.Furnace_heat = Constraint(m.T, rule=furnace_heat_rule)

def furnace_cap_rule(m, t):
    return m.H_fur[t] <= m.Furnace_cap
m.Furnace_cap = Constraint(m.T, rule=furnace_cap_rule)

# (4) Heat-driven chiller/boiler: cooling from available furnace heat with efficiency and
capacity
def cb_cooling_cap_rule(m, t):
    return m.C_cb[t] <= m.CB_cap
m.CB_cap_con = Constraint(m.T, rule=cb_cooling_cap_rule)

def cb_heat_link_rule(m, t):
    # Cooling produced cannot exceed eta_hc times furnace heat (if modeled as absorption
chiller using furnace heat)
    return m.C_cb[t] <= m.eta_hc * m.H_fur[t]
m.CB_heat_link = Constraint(m.T, rule=cb_heat_link_rule)

# (5) EHP performance and mode limits
def ehp_performance_rule(m, t):
    # Total thermal output (heat + cooling) equals COP * electric input
    return m.H_ehp[t] + m.C_ehp[t] == m.COP * m.P_ehp[t]
m.EHP_perf = Constraint(m.T, rule=ehp_performance_rule)

def ehp_heating_bounds_rule(m, t):
    return m.H_min * m.I_h[t] <= m.H_ehp[t] <= m.H_max * m.I_h[t]
m.EHP_heat_bounds = Constraint(m.T, rule=ehp_heating_bounds_rule)

def ehp_cooling_bounds_rule(m, t):
    return m.C_min * m.I_c[t] <= m.C_ehp[t] <= m.C_max * m.I_c[t]
m.EHP_cool_bounds = Constraint(m.T, rule=ehp_cooling_bounds_rule)

def ehp_mode_exclusive_rule(m, t):
```

```
    return m.I_h[t] + m.I_c[t] <= 1
m.EHP_mode_excl = Constraint(m.T, rule=ehp_mode_exclusive_rule)

# (6) Heat balance
def heat_balance_rule(m, t):
    # Heat demand met by CHP heat + Furnace heat + EHP heat (note: some furnace heat may
be diverted to CB; we already limited CB by furnace heat)
    return m.H_chp[t] + m.H_fur[t] + m.H_ehp[t] >= m.Dh[t]
m.HeatBalance = Constraint(m.T, rule=heat_balance_rule)

# (7) Cooling balance
def cooling_balance_rule(m, t):
    return m.C_ehp[t] + m.C_cb[t] >= m.Dc[t]
m.CoolBalance = Constraint(m.T, rule=cooling_balance_rule)

# (8) ESS dynamics and limits
def soc_dynamics_rule(m, t):
    if t == 1:
        return m.SOC[t] == m.SOC0 + (m.eta_c * m.E_ch[t] - (1/m.eta_d) * m.E_dch[t]) *
m.dt
    else:
        return m.SOC[t] == m.SOC[t-1] + (m.eta_c * m.E_ch[t] - (1/m.eta_d) * m.E_dch[t])
* m.dt
m.SOC_dyn = Constraint(m.T, rule=soc_dynamics_rule)

def soc_bounds_rule(m, t):
    return (m.SOCmin, m.SOC[t], m.SOCmax)
m.SOC_bounds = Constraint(m.T, rule=soc_bounds_rule)

def charge_power_bounds_rule(m, t):
    return m.E_ch[t] <= m.Ech_max * m.I_ch[t]
m.ChargePwrBound = Constraint(m.T, rule=charge_power_bounds_rule)

def discharge_power_bounds_rule(m, t):
    return m.E_dch[t] <= m.Edch_max * m.I_dch[t]
m.DischargePwrBound = Constraint(m.T, rule=discharge_power_bounds_rule)

def ess_exclusive_rule(m, t):
    return m.I_ch[t] + m.I_dch[t] <= 1
m.ESS_exclusive = Constraint(m.T, rule=ess_exclusive_rule)

# Optional terminal SOC = initial SOC to avoid end effects
def terminal_soc_rule(m):
    return m.SOC[24] == m.SOC0
m.SOC_terminal = Constraint(rule=terminal_soc_rule)

# ------------------ Solve ------------------
def try_solve(model):
    for cand in ["glpk", "cbc", "highs"]:
        try:
            solver = SolverFactory(cand)
            if solver.available():
                res = solver.solve(model, tee=False)
                return cand, res
        except Exception:
            pass
    return None, None
```

```
if __name__ == "__main__":
    solver_name, results = try_solve(m)
    if solver_name is None:
        print("No MILP solver found. Model built successfully, but not solved.")
        print("Install a solver such as GLPK (glpsol), CBC, or HiGHS and re-run.")
    else:
        print(f"Solved with {solver_name}. Objective value: ${value(m.Obj):,.2f}")
        # Print a compact KPI summary
        tot_grid = sum(value(m.P_grid[t]) for t in m.T)
        tot_gas = sum(value(m.G_chp[t] + m.G_fur[t]) for t in m.T)
        avg_soc = sum(value(m.SOC[t]) for t in m.T) / 24.0
        print(f"Total grid import [MWh]: {tot_grid:.3f}")
        print(f"Total gas use [MWh]: {tot_gas:.3f}")
        print(f"Average SOC [MWh]: {avg_soc:.2f}")
        # Write a CSV of hourly dispatch (optional)
        try:
            import csv
            with open("dispatch_results.csv", "w", newline="") as f:
                w = csv.writer(f)
                header =
["t","P_grid","P_chp","H_chp","G_chp","G_fur","H_fur","C_cb","P_ehp","H_ehp","C_ehp","E_c
h","E_dch","SOC"]
                w.writerow(header)
                for t in T:
                    w.writerow([t,
                                value(m.P_grid[t]), value(m.P_chp[t]), value(m.H_chp[t]),
                                value(m.G_chp[t]), value(m.G_fur[t]), value(m.H_fur[t]),
                                value(m.C_cb[t]), value(m.P_ehp[t]), value(m.H_ehp[t]),
                                value(m.C_ehp[t]), value(m.E_ch[t]), value(m.E_dch[t]),
                                value(m.SOC[t])])
            print("Saved hourly dispatch to dispatch_results.csv")
        except Exception as e:
            print("Could not write dispatch_results.csv:", e)
```