

0. Imports and Setting up Anthropic API Client

```
from google.colab import drive
```

```
drive.mount('/content/drive')
```

Mounted at /content/drive

```
!pip install python-dotenv
```

```
import os
import dotenv
```

```
dotenv.load_dotenv('/content/drive/MyDrive/.env')
```

Collecting python-dotenv
 Downloading python_dotenv-1.0.1-py3-none-any.whl (19 kB)
 Installing collected packages: python-dotenv
 Successfully installed python-dotenv-1.0.1
 True

```
# Load Prompts and Problem Description
```

```
prompt1_path = '/content/drive/MyDrive/Thesis/Prompts/Prompt1_MathematicalModel.txt'
```

```
prompt2_path = '/content/drive/MyDrive/Thesis/Prompts/Prompt2_PyomoCode.txt'
```

```
problem_desc_path = '/content/drive/MyDrive/Thesis/ProblemDescriptions/NL/NL4.txt'
```

```
prompt1_file = open(prompt1_path, "r")
```

```
prompt2_file = open(prompt2_path, "r")
```

```
problem_desc_file = open(problem_desc_path, "r")
```

```
prompt1 = prompt1_file.read()
```

```
print("Prompt 1:\n", prompt1)
```

```
prompt2 = prompt2_file.read()
```

```
print("Prompt 2:\n", prompt2)
```

```
problem_desc = problem_desc_file.read()
```

```
print("Problem Description:\n", problem_desc)
```

Prompt 1:
 Please write a mathematical optimization model for this problem. Include parameters, decision variables, the objective
 Prompt 2:
 Please write a python pyomo code for this optimization problem.
 Use sample data where needed.
 Indicate where you use sample data.
 Problem Description:
 We are looking at an alkylation process which will include the following 10 variables: olefin feed (barrels per day), i
 We want to maximize the daily profit of this alkylation process.
 The profit is defined as the revenue generated from the alkylate yield multiplied with the motor octane number, minus th
 Relationships in terms of other variables for alkylate yield, motor octane number, acid dilution factor, and F-4 perform
 This regression estimate can deviate in both directions from true value of these variables by 2, 1, 5 and 10 percent, re
 Alkylate yield is a function of olefin feed and external isobutane-to-olefine yield. Alkalyte yield equals the amount of
 The motor octane number is derived from the external isobutane-to-olefin ratio and the acid strength. The motor octane n
 The acid dilution factor is calculated based on the F-4 performance number. The acid dillution factor is expressed as 35
 Lastly, the F-4 performance number depends on the motor octane number. F-4 performance number is calculated as -133 plus
 There are some additional constraints imposed by the nature of the chemical process.
 Each variable has a lower and an upper bound.
 The external isobutane-to-olefin ratio needs to equal the ratio of isobutane recycle plus isobutane makeup to olefin fee
 The acid strength needs to equal the ratio of 93000 times acid addition rate to acid addition rate multiplied by acid di
 Lastly, 1.22 alkylate yield needs to be equal to the combined olefin feed and isobutane makeup.

```
!pip install anthropic
```

Collecting anthropic
 Downloading anthropic-0.26.0-py3-none-any.whl (877 kB)
 877.7/877.7 kB 12.0 MB/s eta 0:00:00
 Requirement already satisfied: anyio<5,>=3.5.0 in /usr/local/lib/python3.10/dist-packages (from anthropic) (3.7.1)
 Requirement already satisfied: distro<2,>=1.7.0 in /usr/lib/python3/dist-packages (from anthropic) (1.7.0)
 Collecting httpx<1,>=0.23.0 (from anthropic)
 Downloading httpx-0.27.0-py3-none-any.whl (75 kB)
 75.6/75.6 kB 9.8 MB/s eta 0:00:00
 Requirement already satisfied: pydantic<3,>=1.9.0 in /usr/local/lib/python3.10/dist-packages (from anthropic) (2.7.1)
 Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-packages (from anthropic) (1.3.1)
 Requirement already satisfied: tokenizers>=0.13.0 in /usr/local/lib/python3.10/dist-packages (from anthropic) (0.19.1)
 Requirement already satisfied: typing-extensions<5,>=4.7 in /usr/local/lib/python3.10/dist-packages (from anthropic) (4.
 Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist-packages (from anyio<5,>=3.5.0->anthropi
 Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio<5,>=3.5.0->anthropi

```
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from httpx<1,>=0.23.0->anthropic) (2023.11.17)
Collecting httpcore==1.* (from httpx<1,>=0.23.0->anthropic)
  Downloading httpcore-1.0.5-py3-none-any.whl (77 kB)
    77.9/77.9 kB 9.9 MB/s eta 0:00:00
Collecting h11<0.15,>=0.13 (from httpcore==1.*->httpx<1,>=0.23.0->anthropic)
  Downloading h11-0.14.0-py3-none-any.whl (58 kB)
    58.3/58.3 kB 7.3 MB/s eta 0:00:00
Requirement already satisfied: annotated-types<0.4.0 in /usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1.9.0) (0.4.0)
Requirement already satisfied: pydantic-core==2.18.2 in /usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1.9.0) (2.18.2)
Requirement already satisfied: huggingface-hub<1.0,>=0.16.4 in /usr/local/lib/python3.10/dist-packages (from tokenizers<0.15.0,>=0.13.0->anthropic) (0.16.4)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.16.4->tokenizers) (3.12.2)
Requirement already satisfied: fsspec<2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.16.4->tokenizers) (2023.5.0)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.16.4->tokenizers) (2.31.0)
Requirement already satisfied: tqdm<4.42.1 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.16.4->tokenizers) (4.64.1)
Requirement already satisfied: pyyaml<5.1 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.16.4->tokenizers) (6.0.1)
Requirement already satisfied: packaging<20.9 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.16.4->tokenizers) (23.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hub) (3.3.2)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hub) (2.0.7)
Installing collected packages: h11, httpcore, httpx, anthropic
Successfully installed anthropic-0.26.0 h11-0.14.0 httpcore-1.0.5 httpx-0.27.0
```

```
# Importing Anthropic & Setting Headers
import anthropic
```


```
client = anthropic.Anthropic(
    # defaults to os.environ.get("ANTHROPIC_API_KEY")
    api_key=os.environ.get("ANTHROPIC_API_KEY"),
)
```

1. Prompt 1: Create Mathematical Model

```
message = client.messages.create(
    model="claude-3-opus-20240229",
    system=prompt1,
    max_tokens=4096,
    messages=[
        {"role": "user", "content": problem_desc}
    ]
)
```

```
response_p1 = message.content[0].text
```

```
# Print response
print(response_p1)
```

 Thank you for providing the problem description. Let's formulate the mathematical optimization model for the alkylation

Parameters:

- Let the revenue per barrel of alkylate yield be denoted as R.
- Let the cost per barrel of olefin feed be denoted as C_OF.
- Let the cost per barrel of isobutane recycle be denoted as C_IR.
- Let the cost per thousand pounds of acid addition rate be denoted as C_AAR.
- Let the cost per barrel of isobutane makeup be denoted as C_IM.

Decision Variables:

- Let OF be the olefin feed (barrels per day).
- Let IR be the isobutane recycle (barrels per day).
- Let AAR be the acid addition rate (thousands of pounds per day).
- Let AY be the alkylate yield (barrels per day).
- Let IM be the isobutane makeup (barrels per day).
- Let AS be the acid strength (weight percent).
- Let MON be the motor octane number.
- Let EIO be the external isobutane-to-olefin ratio.
- Let ADF be the acid dilution factor.
- Let F4 be the F-4 performance number.

Objective Function:

Maximize Daily Profit = $R * AY * MON - (C_OF * OF + C_IR * IR + C_AAR * AAR + C_IM * IM)$

Constraints:

1. Alkylate Yield: $0.98 * (1.12 * OF + 0.13167 * OF * EIO - 0.00667 * OF * EIO^2) \leq AY \leq 1.02 * (1.12 * OF + 0.13167 * OF * EIO^2)$
2. Motor Octane Number: $0.99 * (86.35 + 1.098 * EIO - 0.038 * EIO^2 + 0.325 * (AS - 89)) \leq MON \leq 1.01 * (86.35 + 1.098 * EIO - 0.038 * EIO^2 + 0.325 * (AS - 89))$
3. Acid Dilution Factor: $0.95 * (35.82 - 0.222 * F4) \leq ADF \leq 1.05 * (35.82 - 0.222 * F4)$
4. F-4 Performance Number: $0.9 * (-133 + 3 * MON) \leq F4 \leq 1.1 * (-133 + 3 * MON)$
5. External Isobutane-to-Olefin Ratio: $EIO = (IR + IM) / OF$
6. Acid Strength: $AS = (93000 * AAR) / (AAR * ADF + 1000 * AAR)$
7. Material Balance: $1.22 * AY = OF + IM$
8. Variable Bounds:
 - $Lower_Bound_i \leq Variable_i \leq Upper_Bound_i$, for $i = 1, 2, \dots, 10$ (for all variables)

The objective is to maximize the daily profit, which is calculated as the revenue generated from the alkylate yield mult

The constraints include the regression formulas for alkylate yield, motor octane number, acid dilution factor, and F-4 p

✓ 2. Prompt 2: Write Code based on Mathematical Model

```
message2 = client.messages.create(
    model="claude-3-opus-20240229",
    system=prompt2,
    max_tokens=4069,
    messages=[
        {"role": "user", "content": response_p1}
    ]
)

response_p2 = message2.content[0].text

# Print response
print(response_p2)

# Define the constraints
model.alkylate_yield_lower = Constraint(expr=0.98 * (1.12 * model.OF + 0.13167 * model.OF * model.EIO -
    0.00667 * model.OF * model.EIO ** 2) <= model.AY)
model.alkylate_yield_upper = Constraint(expr=model.AY <= 1.02 * (1.12 * model.OF + 0.13167 * model.OF * model.EIO -
    0.00667 * model.OF * model.EIO ** 2))
model.motor_octane_lower = Constraint(expr=0.99 * (86.35 + 1.098 * model.EIO - 0.038 * model.EIO ** 2 +
    0.325 * (model.AS - 89)) <= model.MON)
model.motor_octane_upper = Constraint(expr=model.MON <= 1.01 * (86.35 + 1.098 * model.EIO - 0.038 * model.EIO ** 2 +
    0.325 * (model.AS - 89)))
model.acid_dilution_lower = Constraint(expr=0.95 * (35.82 - 0.222 * model.F4) <= model.ADF)
model.acid_dilution_upper = Constraint(expr=model.ADF <= 1.05 * (35.82 - 0.222 * model.F4))
model.f4_performance_lower = Constraint(expr=0.9 * (-133 + 3 * model.MON) <= model.F4)
model.f4_performance_upper = Constraint(expr=model.F4 <= 1.1 * (-133 + 3 * model.MON))
model.external_isobutane_ratio = Constraint(expr=model.EIO == (model.IR + model.IM) / model.OF)
model.acid_strength = Constraint(expr=model.AS == (93000 * model.AAR) / (model.AAR * model.ADF + 1000 * model.AAR))
model.material_balance = Constraint(expr=1.22 * model.AY == model.OF + model.IM)

# Define variable bounds (using sample data)
model.OF.setlb(5000)
model.OF.setub(10000)
model.IR.setlb(2000)
model.IR.setub(5000)
model.AAR.setlb(1)
model.AAR.setub(5)
model.AY.setlb(4000)
model.AY.setub(8000)
model.IM.setlb(1000)
model.IM.setub(3000)
model.AS.setlb(85)
model.AS.setub(95)
model.MON.setlb(90)
model.MON.setub(100)
model.EIO.setlb(1)
model.EIO.setub(3)
model.ADF.setlb(30)
model.ADF.setub(40)
model.F4.setlb(100)
model.F4.setub(200)

# Solve the optimization problem
SolverFactory('ipopt').solve(model)

# Print the optimal solution
print("Optimal Daily Profit: ", model.profit())
print("Olefin Feed (OF): ", model.OF())
print("Isobutane Recycle (IR): ", model.IR())
print("Acid Addition Rate (AAR): ", model.AAR())
print("Alkylate Yield (AY): ", model.AY())
print("Isobutane Makeup (IM): ", model.IM())
print("Acid Strength (AS): ", model.AS())
print("Motor Octane Number (MON): ", model.MON())
print("External Isobutane-to-Olefin Ratio (EIO): ", model.EIO())
print("Acid Dilution Factor (ADF): ", model.ADF())
print("F-4 Performance Number (F4): ", model.F4())
...
```

Note: The sample data for parameter values and variable bounds are provided for illustrative purposes. You should repl

✓ 4. Input Problem Data and Test Model Code

```
# Installing pyomo nd solver
!pip install -q pyomo
!wget -N -q "https://matematica.unipv.it/gualandi/solvers/ipopt-linux64.zip"
!unzip -o -q ipopt-linux64
```

12.8/12.8 MB 58.1 MB/s eta 0:00:00

49.6/49.6 kB 5.2 MB/s eta 0:00:00

```
from pyomo.environ import *
```

```
# Create the model
model = ConcreteModel()
```

```
# Define the parameters (using sample data)
model.R = Param(initialize=0.63) # Revenue per barrel of alkylate yield
model.C_OF = Param(initialize=5.04) # Cost per barrel of olefin feed
model.C_IR = Param(initialize=0.035) # Cost per barrel of isobutane recycle
model.C_AAR = Param(initialize=10) # Cost per thousand pounds of acid addition rate
model.C_IM = Param(initialize=3.36) # Cost per barrel of isobutane makeup
```

```
# Define the decision variables
model.OF = Var(domain=NonNegativeReals) # Olefin feed (barrels per day)
model.IR = Var(domain=NonNegativeReals) # Isobutane recycle (barrels per day)
model.AAR = Var(domain=NonNegativeReals) # Acid addition rate (thousands of pounds per day)
model.AY = Var(domain=NonNegativeReals) # Alkylate yield (barrels per day)
model.IM = Var(domain=NonNegativeReals) # Isobutane makeup (barrels per day)
model.AS = Var(domain=NonNegativeReals) # Acid strength (weight percent)
model.MON = Var(domain=NonNegativeReals) # Motor octane number
model.EIO = Var(domain=NonNegativeReals) # External isobutane-to-olefin ratio
model.ADF = Var(domain=NonNegativeReals) # Acid dilution factor
model.F4 = Var(domain=NonNegativeReals) # F-4 performance number
```

```
# Define the objective function
model.profit = Objective(expr=model.R * model.AY * model.MON - (model.C_OF * model.OF + model.C_IR * model.IR +
    model.C_AAR * model.AAR + model.C_IM * model.IM), sense=maximize)
```

```
# Define the constraints
model.alkylate_yield_lower = Constraint(expr=0.98 * (1.12 * model.OF + 0.13167 * model.OF * model.EIO -
    0.00667 * model.OF * model.EIO ** 2) <= model.AY)
model.alkylate_yield_upper = Constraint(expr=model.AY <= 1.02 * (1.12 * model.OF + 0.13167 * model.OF * model.EIO -
    0.00667 * model.OF * model.EIO ** 2))
model.motor_octane_lower = Constraint(expr=0.99 * (86.35 + 1.098 * model.EIO - 0.038 * model.EIO ** 2 +
    0.325 * (model.AS - 89)) <= model.MON)
model.motor_octane_upper = Constraint(expr=model.MON <= 1.01 * (86.35 + 1.098 * model.EIO - 0.038 * model.EIO ** 2 +
    0.325 * (model.AS - 89)))
model.acid_dilution_lower = Constraint(expr=0.95 * (35.82 - 0.222 * model.F4) <= model.ADF)
model.acid_dilution_upper = Constraint(expr=model.ADF <= 1.05 * (35.82 - 0.222 * model.F4))
model.f4_performance_lower = Constraint(expr=0.9 * (-133 + 3 * model.MON) <= model.F4)
model.f4_performance_upper = Constraint(expr=model.F4 <= 1.1 * (-133 + 3 * model.MON))
model.external_isobutane_ratio = Constraint(expr=model.EIO == (model.IR + model.IM) / model.OF)
model.acid_strength = Constraint(expr=model.AS == (93000 * model.AAR) / (model.AAR * model.ADF + 1000 * model.AAR))
model.material_balance = Constraint(expr=1.22 * model.AY == model.OF + model.IM)
```

```
# Define variable bounds (using sample data)
model.OF.setlb(0)
model.OF.setub(2000)
model.IR.setlb(0)
model.IR.setub(160000)
model.AAR.setlb(0)
model.AAR.setub(120)
model.AY.setlb(0)
model.AY.setub(5000)
model.IM.setlb(0)
model.IM.setub(2000)
model.AS.setlb(85)
model.AS.setub(93)
model.MON.setlb(90)
model.MON.setub(95)
model.EIO.setlb(3)
model.EIO.setub(12)
model.ADF.setlb(1.2)
model.ADF.setub(4)
model.F4.setlb(145)
model.F4.setub(162)
```

```
# Solve the optimization problem
SolverFactory('ipopt').solve(model)
```

```
# Print the optimal solution
print("Optimal Daily Profit: ", model.profit())
print("Olefin Feed (OF): ", model.OF())
print("Isobutane Recycle (IR): ", model.IR())
```

```
print("Acid Addition Rate (AAR): ", model.AAR())
print("Alkylate Yield (AY): ", model.AY())
print("Isobutane Makeup (IM): ", model.IM())
print("Acid Strength (AS): ", model.AS())
print("Motor Octane Number (MON): ", model.MON())
print("External Isobutane-to-Olefin Ratio (EIO): ", model.EIO())
print("Acid Dilution Factor (ADF): ", model.ADF())
print("F-4 Performance Number (F4): ", model.F4())
```

```
↗ Optimal Daily Profit: 176594.13705361064
Olefin Feed (OF): 2000.0
Isobutane Recycle (IR): 10113.175625405225
Acid Addition Rate (AAR): 0.0
Alkylate Yield (AY): 3278.688557376949
Isobutane Makeup (IM): 2000.0
Acid Strength (AS): 92.88853376059825
Motor Octane Number (MON): 93.79868152416277
External Isobutane-to-Olefin Ratio (EIO): 6.056587762136944
Acid Dilution Factor (ADF): 1.2
F-4 Performance Number (F4): 155.9263365300636
```

✓ 5. Correct The Model Code to Test Mathematical Model (if applicable)