Oil & Gas Economic Model - Code Documentation

Streamlit Application for Economic Analysis of Oil & Gas Projects

---

1. Overview
This Streamlit-based application performs **economic evaluations** for oil and gas field developments. It allows users to:

- Upload production, cost, and makeup gas data.
- Calculate **Net Present Value (NPV)**, **Profitability Index (CPI)**, and **Cash Flow**.
- Compare scenarios with **sensitivity analysis** on key variables (oil price, discount rate, costs).

---

2. Key Features

Feature	Description
Three-Tab Layout	Inputs & Results, Calculation Details, Sensitivity Analysis.
Dynamic Calculations	Adjusts for inflation, discounting, and operating efficiency.
Sensitivity Analysis	Tests Low/Mid/High cases for oil price, discount rate, and costs.
Makeup Gas Scenarios	Compares economics with/without makeup gas costs.
Visualizations	Generates cash flow charts and summary tables.

---

3. Workflow
Tab 1: Inputs & Results

1. **File Uploads**:
    - Production Data File: Oil, gas, and condensate volumes by year.
    - Cost & Drilling Schedule: CAPEX (wells, facilities, workovers).
    - Make-up Gas Schedule: Gas availability for injection.
2. **User Inputs**:
    - Prices (/bbl,/*bbl*,/MMSCF), inflation rates, discount rate, well costs.
    - Toggle for fixed/escalating prices.
3. **Calculations**:
    - **Revenue**: Oil + Condensate + Gas sales, adjusted for inflation.
    - **CAPEX**: Drilling, facilities, workovers (escalated for inflation).
    - **OPEX**: Operating costs + makeup gas (if applicable).
    - **NPV**: Discounted net cash flow.
4. **Outputs**:
    - Key metrics (NPV, PIR, CPI).
    - Cash flow chart (Revenue, CAPEX, OPEX).

Tab 2: Calculation Details

- Displays intermediate dataframes for:
  - CAPEX breakdown (wells, facilities).
  - OPEX (BOE-based costs, makeup gas).
  - Revenue and Net Cash Flow.

Tab 3: Sensitivity Analysis

1. **User Defines Bounds**:
   - Low/High oil price, discount rate, cost per BOE.
2. **Two Scenarios**:
   - **No Makeup Gas Cost**: makeup_gas_cost = 0.
   - **With Makeup Gas Cost**: Uses user-input cost.
3. **Outputs**:
   - Tables comparing NPV, Revenue, Costs for Low/Mid/High cases.
   - Cash flow plots for each scenario.

---

4. Critical Fixes (Based on User Feedback)

Issue:

- Sensitivity analysis showed identical results for both makeup gas scenarios because OPEX2 MM$ was not recalculated dynamically.

Solution:

Modified the run_case() function to **recompute makeup gas costs per scenario**:

Python

```
def run_case(oil_p, disc_rate, cost_boe, current_makeup_cost_for_case):
    df_temp = df.copy()
    # Recalculate OPEX2 (makeup gas) from scratch
    df_temp["OPEX2 MM$"] = (current_makeup_cost_for_case * makeup_gas_daily_mmscf *
365 * df_temp["Availability"] / 1e3)
    df_temp["Total OPEX MM$"] = df_temp["OPEX1 MM$"] + df_temp["OPEX2 MM$"]
    # ... (rest of calculations)
```

Scenario Execution:

python

```
# Scenario 1: No Makeup Gas Cost
run_case(..., current_makeup_cost_for_case=0)  # Zero cost for gas

# Scenario 2: With Makeup Gas Cost
run_case(..., current_makeup_cost_for_case=makeup_gas_cost)  # User-defined cost
```

---

5. Code Structure

markdown
Copy
Download

```
eco_app_test.py
├── Session State Setup
├── Page Config & Tabs
│   ├── Tab 1: Inputs & Results
│   │   ├── File Uploads
│   │   ├── User Inputs (Prices, Costs, Inflation)
│   │   ├── Calculations (CAPEX, OPEX, Revenue, NPV)
│   │   └── Output (Metrics, Cash Flow Chart)
│   ├── Tab 2: Calculation Details
│   │   └── DataFrames (CAPEX, OPEX, Revenue Breakdowns)
│   └── Tab 3: Sensitivity Analysis
│       ├── Low/Mid/High Case Inputs
│       ├── Scenario Execution (No/With Makeup Gas)
│       └── Output (Tables, Plots, Download)
└── Dynamic Plotting & Error Handling
```

---

6. Usage Instructions

1. **Upload Files**:
    - Ensure Excel files match the expected columns (e.g., Year, Oil Prod STB, Availability).
2. **Run Calculations**:
    - Click "Calculate" in Tab 1 to process data.
3. **Adjust Sensitivity Parameters**:
    - Modify bounds in Tab 3 and click "Run Combined Sensitivity Analysis".
4. **Download Results**:
    - Export sensitivity results as a .txt file.

---

7. Dependencies

- Python 3.8+
- Libraries:

plaintext
Copy
Download

```
streamlit, pandas, numpy, matplotlib, io, os
```

---

8. Notes for Maintenance

- **Column Names**: Ensure uploaded files match expected headers (e.g., Gas Prod SCF).
- **Inflation Logic**: Verify escalation formulas if assumptions change.
- **Session State**: Reset (st.session_state.clear()) if recalculations behave unexpectedly.