Technical Report: Python NPV & IRR Calculator

Anshuman Nema

1 Approach

The objective was to build a robust financial calculator using **numerical analysis** rather than pre-built financial libraries (like `numpy-financial`). The design philosophy focused on:

- **Simplicity:** Using standard Python lists and loops to ensure the code is readable for beginners.
- **Modularity:** Separating the mathematical logic (`npv`, `irr`) from the presentation logic (`analyze_project`), making the code easier to test and modify.
- **Usability:** Prioritizing professional formatting (Rupee symbols, aligned tables) and visual feedback (`matplotlib` graphs) to make the data interpretable.

2 Methodology

2.1 Net Present Value (NPV)

I implemented the standard discrete discounting formula:

$$NPV = \sum_{t=0}^{n} \frac{C_t}{(1+r)^t}$$

This was achieved using a `for` loop with `enumerate()`, allowing us to track both the time period t and the cash flow amount simultaneously.

2.2 Internal Rate of Return (IRR) - The Bisection Method

I utilized the **Bisection Method** for root-finding.

- **Search Interval:** The algorithm searches between -99% and 1000%.
- **Convergence:** It runs for a maximum of 1,000 iterations or until the error is $< 1e^{-6}$.
- **Robust Logic:** Crucially, the algorithm checks for a **sign change** (`npv_low * npv_mid > 0`) rather than a simple positive/negative value. This ensures the solver works for both standard investments (where NPV decreases as rates rise) and financing/loan scenarios (where NPV increases as rates rise).

2.3 Visualization

I utilized `matplotlib` to generate a dual-subplot figure:

- **Cash Flow Diagram:** A bar chart colored dynamically (Green for inflows, Red for outflows) to visualize the project structure.
- **NPV Profile:** A line graph plotting NPV against various discount rates to visually identify the IRR (x-intercept).

3 Key Insights Gained

The "Financing" Trap

During testing, I discovered that standard bisection logic fails for "Financing" projects (like loans). Most textbook algorithms assume NPV slopes downward. By switching to **Sign-Comparison Logic** (Intermediate Value Theorem), I ensured the calculator handles all types of cash flows correctly.

The Importance of Sensitivity Analysis

A single IRR number is often misleading. Implementing the Sensitivity Table (showing NPV at 5%, 10%, 15%) revealed that some projects might have a high IRR but very low total cash value, providing a more complete picture of financial risk.

User Experience in Code

Adding small details—like formatting strings (`:,.2f` for currency) and using `hspace` to prevent graph overlap—significantly improved the readability and "professional feel" of the tool.