Simulador de Juros sobre Reservas: Uma Perspectiva Baseada na Teoria de Kessler com Python

Contexto da Gestão Financeira

A gestão financeira de recursos onerosos de terceiros, notadamente a relação entre reservas de lucros, despesas de juros e empréstimos é crucial para a saúde financeira de uma empresa.

O acompanhamento de perto desses recursos é fundamental para reduzir a dependência e, consequentemente, as despesas com juros.

Envolve também uma gestão eficiente das reservas de lucros como elemento vital para o financiamento de investimentos, para o fortalecimento dos negócios, crescimento e geração de lucros, mas sem a necessidade descontrolada de empréstimos onerosos.

Este artigo, tem como propósito apresentar um simulador financeiro desenvolvido em Python que permita simular e visualizar a relação entre despesas financeiras acumuladas e reservas de lucros restantes ao longo de um período previsto, através da analogia da síndrome de Kessler no impacto financeiro da empresa.

A Teoria de Kessler e as Despesas Financeiras

A Teoria de Kessler ou Síndrome de Kessler foi proposta pelo cientista Donald J. Kessler em 1978 no contexto da astronomia e engenharia especial, especialmente no campo do lixo de detritos espaciais na orbita terrestre, que dado o acúmulo de objetos resultaria em colisões e destruição de satélites.

Assim como esse conceito está aplicado ao espacial sideral onde eventos isolados podem desencadear uma reação em cadeia elevando o risco de colapso do sistema, também podemos visualizá-lo e compará-lo no contexto da saúde financeira das empresas onde o acúmulo de dívidas ou despesas com juros decorrente do capital oneroso de terceiros, contraídos sem controle, podem levar ao caos e/ou a falência da organização.

Contextualizando o Script no Fenômeno de Kessler

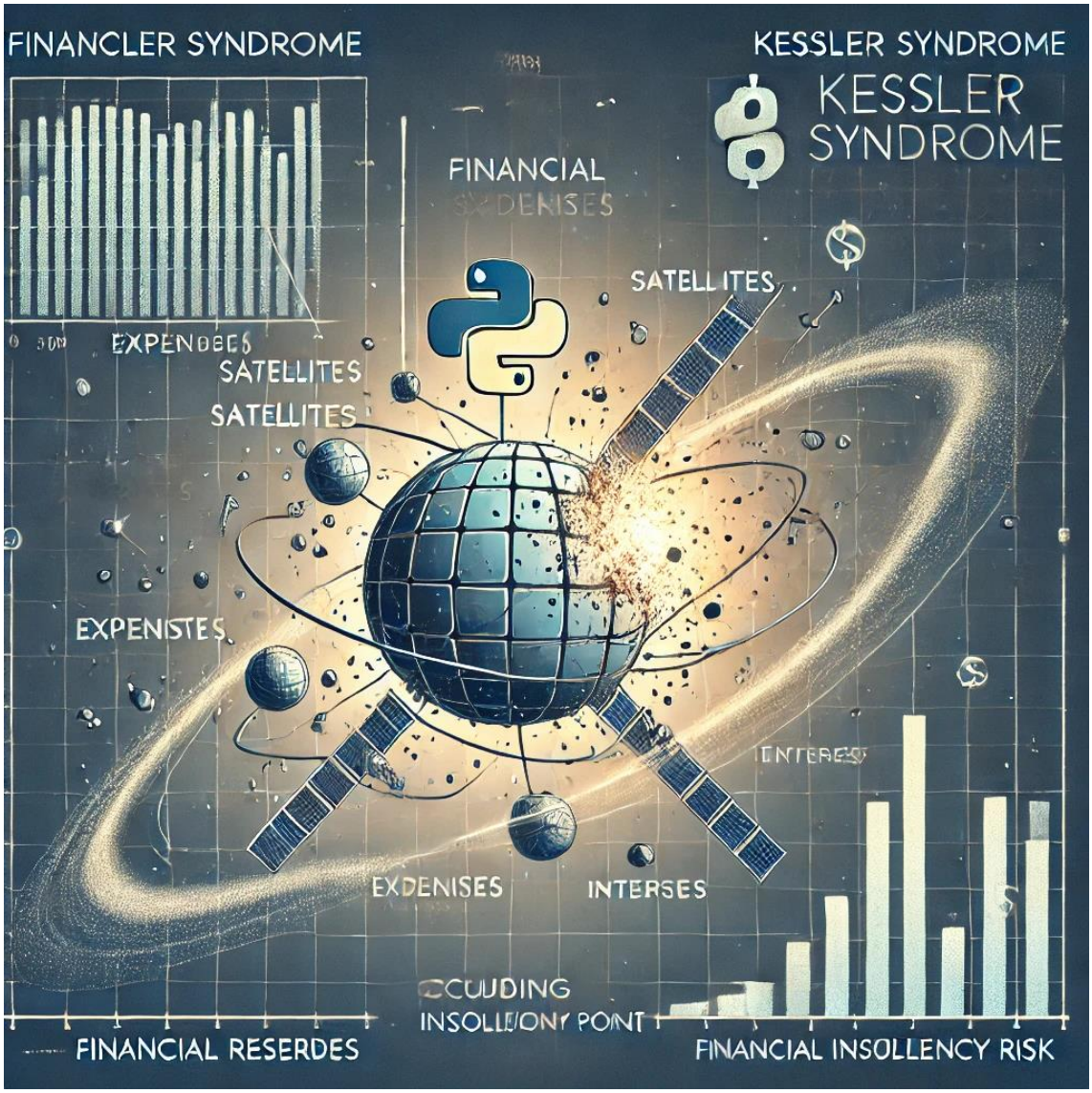
O script desenvolvido tem como objetivo servir como uma ferramenta analítica para identificar o exato ponto de colisão financeira que ocorrerá entre as despesas financeiras acumuladas e as reservas de lucros que restaram, ano após ano, após a dedução daquelas pelo uso indiscriminado de empréstimos onerosos junto a terceiros.

Ou seja, a simulação permitirá visualizar o momento em que as despesas acumulados com juros se tornam tão significativas que colidem diretamente com as reservas disponíveis e, por conseguinte, a capital da empresa a níveis perigosos.

A colisão é demonstrada no ponto de intersecção entre as curvas das linhas representativas das reservas de lucros restantes e as despesas financeiras acumuladas. Isto é, essa intersecção é o

ponto crítico que marca o início de uma possível reação destruidora da capacidade financeira da empresa de honrar seus compromissos.

Analogia da Gestão Financeira com o Fenômeno de Kessler



A analogia principal entre o fenômeno Kessler e o equilíbrio entre as reservas de lucros e as despesas financeiras se dá pela correlação da seguinte forma:

a) Na teoria de Kessler o espaço orbital é o ambiente onde se situam os satélites e outros objetos espaciais. Na gestão dos recursos, as reservas de lucros são equivalentes a esse espaço.
b) As despesas e juros, por sua vez, são os satélites e outros “detritos” ocupando o espaço das reservas financeiras. Assim, como prevê aquela teoria em que o volume crescente e

elevado desses detritos provocará, em algum momento, uma colisão entre eles, o mesmo poderá ocorrer com as despesas financeiras, colidindo umas com as outras, no “espaço” das reservas restantes acumuladas.

c) O momento de colisão dos detritos é, por sua vez, o “momento” em que no gráfico ocorre a intersecção das despesas com as reservas. É o ponto crítico inicial que sinaliza um provável risco de insolvência da empresa.

O ponto de interseção é o principal indicador inspirado no script utilizado para medir o impacto dessa relação entre as despesas e as reservas ao longo do período previsto.

Síndrome de Kessler	Ponto de Colisão Financeiro
Espaço Orbital	Reservas Acumuladas de Lucros
Detritos	Despesas Financeiras e Juros
Colisão de Detritos	Ponto de Colisão (Intersecção das Linhas)

Fonte: o Autor

O Modelo Matemático por Trás do Ponto de Colisão

O papel da simulação proposta no script é calcular e exibir de forma gráfica como as despesas financeiras crescem ao longo do tempo e impactam as reservas de lucros.

A tomada indiscriminada de empréstimos junto a terceiros sem controle do volume do que já se tem contratado, com o script, poderá ser visualizado, fornecendo insights valiosos para prevenir essa situação.

A matemática do ponto de colisão (a intersecção das duas curvas) se dará pelo ajuste das despesas, calculadas como a soma total das despesas ao longo do período dividido por dois (representando os dois principais componentes de custos: juros de empréstimos existentes e novos empréstimos). As reservas de lucros, por sua vez, serão obtidas subtraindo as despesas ajustadas do valor inicial das reservas.

A interseção das linhas azul e vermelha no gráfico simboliza a colisão financeira, permitindo uma análise visual clara de como e quando essa situação ocorre

O Que Faz o Simulador?

De maneira geral, o simulador:

- Recebe dados financeiros, como valores de reservas iniciais, empréstimos existentes e novos, taxas de juros e o período de previsão.
- Calcula como as reservas e as despesas acumuladas evoluem ao longo do tempo.
- Identifica o **ponto de interseção** entre reservas e despesas, onde a situação financeira pode se tornar crítica.
- Gera uma tabela de resultados no terminal, um gráfico ilustrativo e um relatório em PDF contendo o parecer técnico do analista.

Como Funciona?

1. **Entrada de Dados:** O usuário fornece os seguintes valores:
    - Reservas iniciais (R$).
    - Saldo devedor de empréstimos existentes (R$).
    - Taxa de juros dos empréstimos existentes (% ao ano).
    - Valor do novo empréstimo (R$).
    - Taxa de juros do novo empréstimo (% ao ano).
    - Período de previsão (em anos).
    - Nome do analista responsável pelo relatório.
2. **Processamento e Resultados:**
    - Os cálculos são realizados com base nas entradas.
    - Os resultados incluem:
        - Uma **tabela no terminal** mostrando as reservas restantes e despesas acumuladas ao longo do período.
        - Um **gráfico** bidimensional, com a linha azul representando as reservas e a linha vermelha as despesas acumuladas.
        - Um **relatório em PDF**, com uma introdução, dados da simulação e um parecer técnico.

Reservas e Despesas Financeiras

Para entender melhor o propósito do simulador, é importante diferenciar:

- **Reservas Financeiras:** O "caixa" disponível da empresa, usado para cobrir despesas e enfrentar emergências.
- **Despesas Financeiras:** Os custos relacionados a dívidas, como juros e taxas bancárias.

A relação entre reservas e despesas é crucial. Despesas excessivas podem consumir rapidamente as reservas, enquanto reservas inadequadas podem colocar a empresa em risco de insolvência.

O simulador destaca essa relação, permitindo identificar:

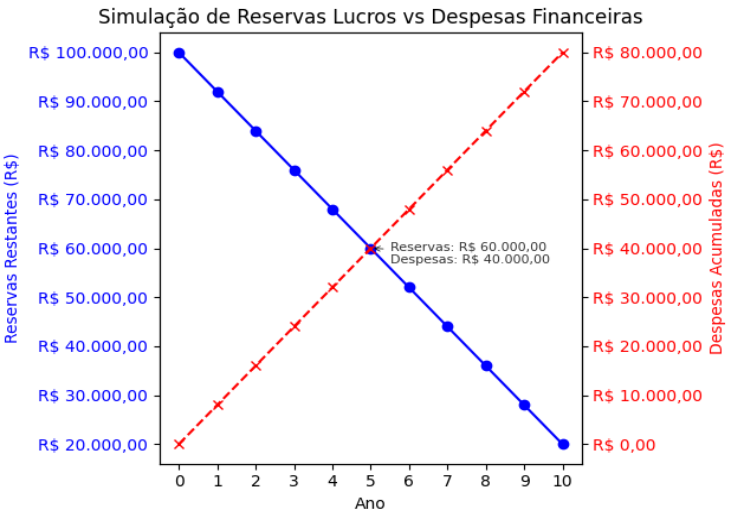
- **Cenários positivos:** onde as reservas são suficientes para cobrir as despesas acumuladas.
- **Cenários críticos:** onde as reservas acabam antes do fim do período analisado.

As Duas Situações: Reservas Positivas e Negativas

O simulador é capaz de prever duas situações distintas:

1. **Reservas Positivas (Cenário Favorável):**
    - As reservas disponíveis são suficientes para cobrir as despesas acumuladas.
    - O gráfico mostra as linhas azul (reservas) e vermelha (despesas) sem se cruzarem ou com reservas ainda positivas no final do período.
    - Parecer técnico: "As reservas disponíveis são suficientes. Recomenda-se monitoramento contínuo."

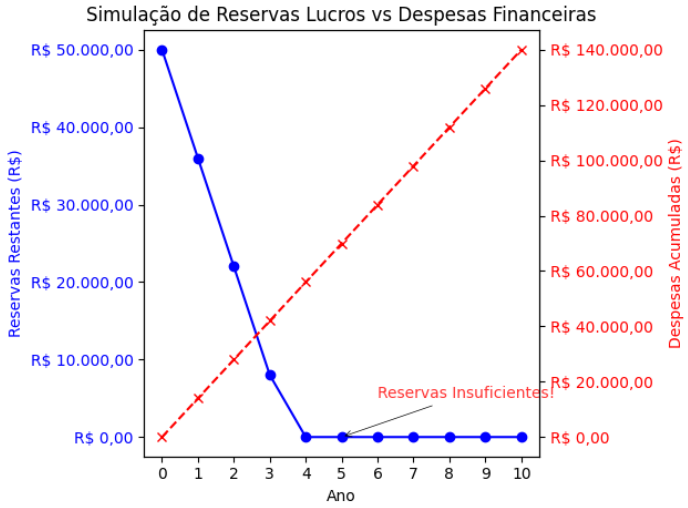
    Exemplo Visual:



2. **Reservas Negativas (Cenário Crítico):**
    - As despesas acumuladas ultrapassam as reservas disponíveis.
    - O gráfico destaca o ponto onde as reservas zeram, indicando "Reservas Insuficientes!".

- Parecer técnico: "As reservas disponíveis são insuficientes. Recomenda-se revisão urgente das finanças."

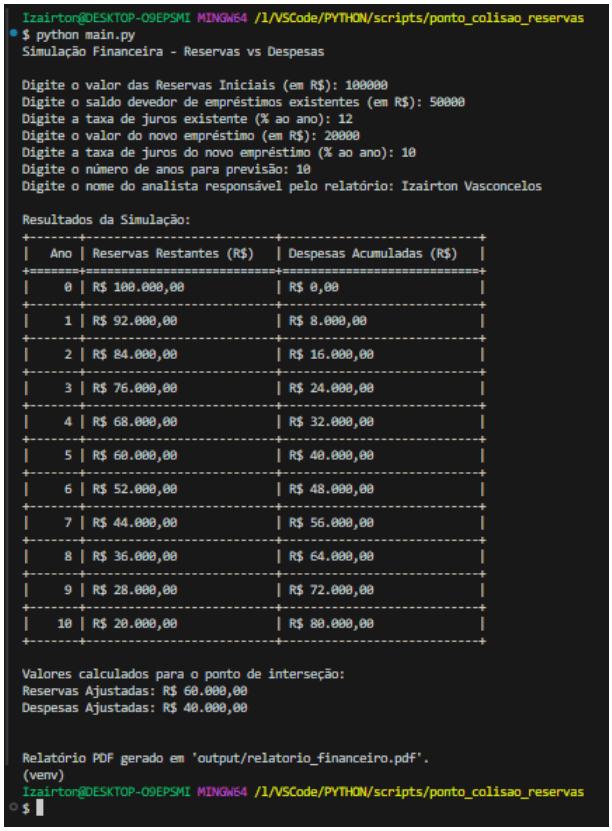
Exemplo Visual:



Como Usar o Simulador?

O processo é simples:

1. **Executar o Script:** Use o comando python main.py no terminal.
2. **Fornecer os Dados:** Preencha as entradas solicitadas no terminal.
3. **Analisar os Resultados:**
   - Uma tabela detalhada será exibida no terminal.
   - Um gráfico ilustrativo será gerado automaticamente.
   - Um relatório PDF será salvo na pasta /output, com o parecer técnico baseado na simulação.



A Importância do Python no Projeto

O uso do Python foi crucial neste projeto:

- Permitiu a automação dos cálculos e das previsões financeiras.
- Facilitou a geração de gráficos e relatórios profissionais.
- Tornou a simulação acessível, rápida e altamente adaptável a diferentes cenários.

Com bibliotecas como matplotlib, pandas, tabulate e fpdf, o Python demonstrou ser uma ferramenta poderosa para análises financeiras complexas, tornando-o indispensável neste contexto.

Conclusão

O simulador de reservas e despesas financeiras provou ser uma ferramenta indispensável para gestores e analistas, permitindo:

- Visualizar o impacto das decisões financeiras ao longo do tempo.
- Identificar situações de risco antes que se tornem críticas.
- Auxiliar na tomada de decisões estratégicas com base em dados claros e precisos.

ANEXO – Os Códigos

a) Simulação.py

```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.ticker import FuncFormatter

# Função para formatar valores como moeda brasileira
def format_currency(value):
    """
    Formata um valor numérico como moeda brasileira (R$).
    """
    return f"R$ {value:,.2f}".replace(",", "X").replace(".",
",").replace("X", ".")

# Função para simular as despesas acumuladas e reservas ao longo dos anos
def simulate_expenses_vs_reserves(initial_reserves, total_expense,
years_to_predict):
    """
    Simula as despesas acumuladas e reservas restantes ao longo dos anos.

    Parâmetros:
        initial_reserves (float): Reservas iniciais.
        total_expense (float): Despesa total anual.
        years_to_predict (int): Número de anos para previsão.

    Retorna:
        pd.DataFrame: DataFrame contendo anos, reservas restantes e
despesas acumuladas.
    """
    reserves = [initial_reserves]
    expenses = []

    for year in range(years_to_predict + 1):
        # Calcula despesas acumuladas
        expenses.append(total_expense * year if year > 0 else 0)
        # Atualiza as reservas remanescentes
        remaining_reserves = max(reserves[-1] - total_expense, 0)
        reserves.append(remaining_reserves)

    return pd.DataFrame({
        "Ano": list(range(years_to_predict + 1)),
        "Reservas Restantes (R$)": reserves[:years_to_predict + 1],
        "Despesas Acumuladas (R$)": expenses
    })

# Função para calcular o ponto de interseção
```

```python
def calculate_intersection_point(df, initial_reserves, years_to_predict):
    """
    Calcula o ponto de interseção entre reservas e despesas acumuladas.

    Parâmetros:
        df (pd.DataFrame): DataFrame contendo os dados da simulação.
        initial_reserves (float): Reservas iniciais.
        years_to_predict (int): Número de anos para previsão.

    Retorna:
        tuple: Reservas e despesas ajustadas no ponto de interseção.
    """
    # Remover formato de moeda brasileira e converter para float
    cumulative_expenses_total = df.loc[years_to_predict, "Despesas
Acumuladas (R$)"]
    if isinstance(cumulative_expenses_total, str):
        cumulative_expenses_total = float(
            cumulative_expenses_total.replace("R$", "").replace(".",
"").replace(",", ".").strip()
        )

    intersection_expenses = cumulative_expenses_total / 2
    intersection_reserves = initial_reserves - intersection_expenses
    return intersection_reserves, intersection_expenses

# Função para gerar o gráfico de Reservas vs Despesas
def generate_expenses_vs_reserves_graph(df, initial_reserves,
years_to_predict):
    """
    Gera o gráfico de Reservas Restantes vs Despesas Acumuladas.

    Parâmetros:
        df (pd.DataFrame): DataFrame contendo os dados da simulação.
        initial_reserves (float): Reservas iniciais.
        years_to_predict (int): Número de anos para previsão.
    """
    fig, ax1 = plt.subplots()

    years = df["Ano"]
    reserves = df["Reservas Restantes (R$)"]
    expenses = df["Despesas Acumuladas (R$)"]

    # Configuração do eixo de Reservas (azul)
    ax1.yaxis.set_major_formatter(FuncFormatter(lambda x, _:
format_currency(x)))
    ax1.plot(years, reserves, 'b-o', label="Reservas Restantes (R$)")
    ax1.set_xlabel("Ano")
    ax1.set_ylabel("Reservas Restantes (R$)", color="blue")
    ax1.tick_params(axis='y', labelcolor="blue")
```

```
    ax1.set_xticks(years)

    # Configuração do eixo de Despesas (vermelho)
    ax2 = ax1.twinx()
    ax2.yaxis.set_major_formatter(FuncFormatter(lambda x, _:
format_currency(x)))
    ax2.plot(years, expenses, 'r--x', label="Despesas Acumuladas (R$)")
    ax2.set_ylabel("Despesas Acumuladas (R$)", color="red")
    ax2.tick_params(axis='y', labelcolor="red")

    # Calcular o ponto de interseção
    intersection_reserves, intersection_expenses =
calculate_intersection_point(df, initial_reserves, years_to_predict)

    # Exibir os valores calculados no terminal
    print(f"\nValores calculados para o ponto de interseção:")
    print(f"Reservas Ajustadas:
{format_currency(intersection_reserves)}")
    print(f"Despesas Ajustadas:
{format_currency(intersection_expenses)}\n")

    # Adicionar anotação no gráfico próximo à interseção
    if intersection_reserves <= 0:
        ax1.annotate(
            "Reservas Insuficientes!",
            xy=(years_to_predict / 2, 0),
            xytext=(years_to_predict / 2 + 1, initial_reserves * 0.1),
            arrowprops=dict(facecolor='red', arrowstyle="->", lw=0.5),
            fontsize=10,
            color="red",
            alpha=0.8
        )
    else:
        ax1.annotate(
            f"Reservas:
{format_currency(intersection_reserves)}\nDespesas:
{format_currency(intersection_expenses)}",
            xy=(years_to_predict / 2, intersection_reserves),
            xytext=(years_to_predict / 2 + 0.5, intersection_reserves -
(intersection_reserves * 0.05)),
            arrowprops=dict(facecolor='black', arrowstyle="->", lw=0.5),
            fontsize=8,
            color="black",
            alpha=0.8
        )

    plt.title("Simulação de Reservas Lucros vs Despesas Financeiras")
    plt.tight_layout()
    plt.show()
```

b) Main.py

```
from src.simulacao import (
    simulate_expenses_vs_reserves,
    calculate_intersection_point,
    generate_expenses_vs_reserves_graph,
    format_currency
)
from gerar_relatorio import generate_pdf_report
from tabulate import tabulate

def parse_float(value):
    try:
        return float(value)
    except ValueError:
        print("Entrada inválida! Por favor, insira um valor numérico.")
        exit()

def main():
    print("Simulação Financeira - Reservas vs Despesas\n")

    # Entrada de dados
    initial_reserves = parse_float(input("Digite o valor das Reservas
Iniciais (em R$): "))
    existing_loans = parse_float(input("Digite o saldo devedor de
empréstimos existentes (em R$): "))
    existing_rate = parse_float(input("Digite a taxa de juros existente
(% ao ano): "))
    new_loan = parse_float(input("Digite o valor do novo empréstimo (em
R$): "))
    new_rate = parse_float(input("Digite a taxa de juros do novo
empréstimo (% ao ano): "))
    years_to_predict = int(input("Digite o número de anos para previsão:
"))
    analista = input("Digite o nome do analista responsável pelo
relatório: ")

    # Calcular despesas totais
    total_expense = (existing_loans * existing_rate / 100) + (new_loan *
new_rate / 100)

    # Simular valores
    df = simulate_expenses_vs_reserves(initial_reserves, total_expense,
years_to_predict)

    # Calcular ponto de interseção
```

```
    intersection_reserves, intersection_expenses =
calculate_intersection_point(df, initial_reserves, years_to_predict)

    # Copiar DataFrame para exibição com formatação
    df_formatted = df.copy()
    df_formatted["Reservas Restantes (R$)"] = df["Reservas Restantes
(R$)"].apply(format_currency)
    df_formatted["Despesas Acumuladas (R$)"] = df["Despesas Acumuladas
(R$)"].apply(format_currency)

    # Mostrar resultados no terminal
    print("\nResultados da Simulação:")
    print(tabulate(df_formatted, headers="keys", tablefmt="grid",
showindex=False, numalign="right"))

    # Gerar gráfico (usando o DataFrame original)
    generate_expenses_vs_reserves_graph(df, initial_reserves,
years_to_predict)

    # Gerar relatório PDF (usando o DataFrame formatado)
    generate_pdf_report(df_formatted, intersection_reserves,
intersection_expenses, analista)
    print("\nRelatório PDF gerado em 'output/relatorio_financeiro.pdf'.")

if __name__ == "__main__":
    main()
```

c) gerar_relatorio.py

```
from fpdf import FPDF
from datetime import datetime

class PDFReport(FPDF):
    def header(self):
        self.set_font('Arial', 'B', 12)
        self.cell(0, 10, "Relatório Financeiro - Simulação de Reservas vs
Despesas", border=False, ln=True, align='C')
        self.ln(5)

    def footer(self):
        self.set_y(-15)
        self.set_font('Arial', 'I', 8)
        self.cell(0, 10, f'Gerado em: {datetime.now().strftime("%d/%m/%Y
%H:%M:%S")}', align='R')
```

```
def generate_pdf_report(df, intersection_reserves, intersection_expenses,
analista):
    """
    Gera o relatório financeiro em PDF.

    Parâmetros:
        df (pd.DataFrame): DataFrame com os dados da simulação.
        intersection_reserves (float): Reservas ajustadas no ponto de
interseção.
        intersection_expenses (float): Despesas ajustadas no ponto de
interseção.
        analista (str): Nome do analista responsável pelo relatório.
    """
    pdf = PDFReport()
    pdf.add_page()

    # Cabeçalho com data e local
    local = "Ponta Grossa - PR"
    data = datetime.now().strftime("%d de %B de %Y")
    pdf.set_font("Arial", size=12)
    pdf.cell(0, 10, f"{local}, {data}", ln=True, align="R")
    pdf.ln(10)

    # Endereçamento
    pdf.set_font("Arial", "B", size=12)
    pdf.cell(0, 10, "À", ln=True)
    pdf.cell(0, 10, "Empresa XYZ", ln=True)
    pdf.cell(0, 10, "Sr. Administrador", ln=True)
    pdf.ln(10)

    # Introdução
    pdf.set_font("Arial", size=12)
    pdf.multi_cell(0, 10, txt=(
        "Conforme sua solicitação, apresentamos o relatório com base na
análise financeira "
        "das reservas e despesas acumuladas, considerando as condições
fornecidas. "
        "Abaixo, seguem os resultados da simulação e o parecer técnico."
    ))
    pdf.ln(10)

    # Resultados do Ponto de Interseção
    pdf.set_font("Arial", "B", size=12)
    pdf.cell(0, 10, "Ponto de Interseção Calculado:", ln=True)
    pdf.set_font("Arial", size=12)
    pdf.cell(0, 10, f"Reservas Ajustadas: R$
{intersection_reserves:,.2f}".replace(",", "X").replace(".",
",").replace("X", "."), ln=True)
```

```
    pdf.cell(0, 10, f"Despesas Ajustadas: R$
{intersection_expenses:,.2f}".replace(",", "X").replace(".",
",").replace("X", "."), ln=True)
    pdf.ln(10)

    # Resultados Ano a Ano
    pdf.set_font("Arial", "B", size=12)
    pdf.cell(0, 10, "Resultados Ano a Ano:", ln=True)
    pdf.ln(5)

    pdf.set_font("Arial", size=10)
    for index, row in df.iterrows():
        pdf.cell(0, 10, f"Ano {row['Ano']} - Reservas: {row['Reservas
Restantes (R$)']} - Despesas: {row['Despesas Acumuladas (R$)']}",
ln=True)

    pdf.ln(10)

    # Parecer Técnico
    pdf.set_font("Arial", "B", size=12)
    pdf.cell(0, 10, "Parecer Técnico:", ln=True)
    pdf.set_font("Arial", size=12)

    if intersection_reserves <= intersection_expenses:
        pdf.multi_cell(0, 10, txt=(
            "As reservas disponíveis estão em risco de se tornarem
insuficientes em relação às despesas acumuladas. "
            "Recomenda-se revisar os planos financeiros, ajustar o
orçamento e evitar novos compromissos financeiros."
        ))
    else:
        pdf.multi_cell(0, 10, txt=(
            "As reservas disponíveis são suficientes para cobrir as
despesas acumuladas no período analisado. "
            "Recomenda-se manter o monitoramento contínuo e garantir a
estabilidade financeira."
        ))

    # Assinatura do Analista
    pdf.ln(20)
    pdf.set_font("Arial", "B", size=12)
    pdf.cell(0, 10, "Atenciosamente,", ln=True)
    pdf.cell(0, 10, analista, ln=True)

    # Salvar o PDF
    pdf.output("output/relatorio_financeiro.pdf")
```

Siga-me no LinkedIn: www.linkedin.com/comm/mynetwork/discovery-see-all?usecase=PEOPLE_FOLLOWS&followMember=izairton-oliveira-de-vasconcelos-a1916351

Minha Newsletter, o link para assinar:
https://www.linkedin.com/build-relation/newsletter-follow?entityUrn=7287106727202742273

Link do artigo no Linkedin:
https://www.linkedin.com/pulse/simulador-de-juros-sobre-reservas-uma-perspectiva-na-com-vasconcelos-rlrvf

https://github.com/IOVASCON/ponto_colisor_reservas.git