Simulador da Alta do Dólar e o Impacto nas Reservas Cambiais

Momento de Incertezas

Vivemos um momento de grande apreensão no cenário econômico brasileiro. As constantes altas do dólar e a queima regular do estoque de nossas reservas cambiais pelo Banco Central, acendeu não só o sinal vermelho do perigo da inflação como também os sinais da desordem e insegurança.

Diante desse cenário, este artigo tem como objetivo apresentar um script em Python que simulará diferentes situações da alta do dólar e da queima das reservas cambiais pelo Governo Federal.

A ideia é bem simples. A partir das recentes altas do dólar e de algumas estratégias contracionistas prováveis do governo, o script

retornará uma simulação indicando o quanto de estoque será queimado para conter a alta da moeda americana.
A lógica aplicada no script também poderá servir para simular eventos de capitais de giro onerosos frente aos juros pagos a terceiros pela redução de reservas financeiras. O script em Python demonstrará através de tabelas, referências visuais e textos explicativos a problemática discutida.

O Cenário Atual e o Modelo Matemático

Um dos principais sinais de que uma economia vai indo de mal a pior ou pelo menos apontado para o abismo é quando um governo resolve queimar suas reservas cambiais para conter a alta de outra moeda, em nosso caso o dólar, sem se importar com a verdadeira causa.

O ponto chave aparece quando o Banco Central intervém no mercado para conter a alta do dólar utilizando para isso a torra de dólares reduzindo nossas reservas cambiais.

Usaremos nesse modelo a regressão linear para definir as cotações futuras da moeda considerando as altas passadas dos últimos dias. Em seguida, utilizaremos de diferentes estratégias do governo para simular a evolução das reservas cambiais e sua queima no mercado. Além disso, ajustaremos o valor do dólar com base na estratégia e no sentimento de mercado, conceito que adotaremos como sendo outros fatores externos influenciadores do câmbio.

Criação de um Script em Python

O desenvolvimento do modelo para simular esse cenário partiu inicialmente da necessidade de prever os valores futuros do dólar, dado as constantes altas apresentadas, e o impacto nas reservas cambiais do Brasil.

Como as reservas estão sendo queimadas continuamente para acalmar o mercado e a fuga de capitais, o modelo também incluiu prováveis estratégias que poderiam justificar a queima dos dólares por parte do governo federal. Essas estratégias foram definidas como: Moderada, Agressiva, Inatividade e Padrão.

Para não ficar apenas no campo governamental, optou-se também pela inclusão de um fator que evidenciasse as interferências externas diretamente no mercado, refletindo neste, um estado otimista ou pessimista diante da evolução do dólar e da eficácia da queima de reservas.

O trecho do código que sinaliza esse entendimento está evidenciado através dos valores informados pelo usuário na interface gráfica do projeto, reproduzido a seguir:

```
# Configurações iniciais do script
dollar_values = [6.00, 6.05, 6.10, 6.15, 6.20] # Valores históricos do dólar
initial_reserves = 170 # Reservas iniciais (em bilhões de dólares)
burn_rate = 1.7 # Taxa diária de queima de reservas (em bilhões de dólares)
days_to_predict = 10 # Número de dias para previsão
strategies = ['moderada', 'agressiva', 'inatividade', 'padrão'] # Estratégias simuladas
market_sentiment = -0.5 # Sentimento de mercado: -1 (piora), 0 (neutro), 1 (melhora)
```

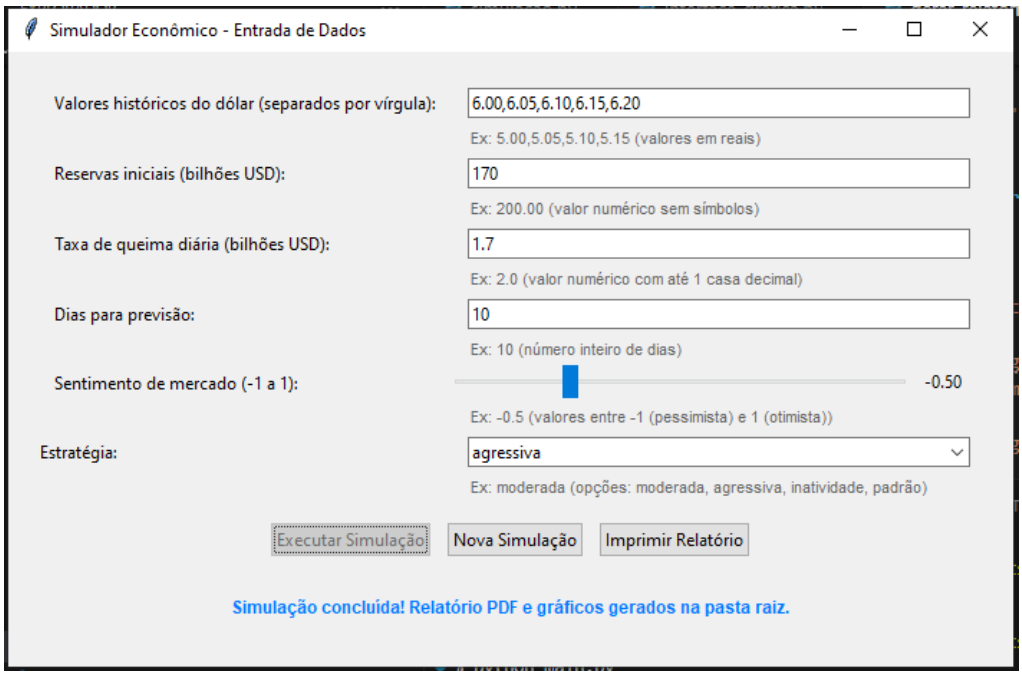


Imagem da Interface Gráfica.

Por fim, o script retornará cada estratégia com um texto explicativo da discrepância entre a alta do dólar e a torra de reservas cambiais, em consonância com o sentimento do mercado de piora, melhora ou neutralidade.

O Funcionamento do Script em Python

O projeto é estruturado de maneira modular para garantir a eficiência, legibilidade e reutilização de código. Ele combina cálculos matemáticos avançados, uma interface gráfica amigável e geração de relatórios profissionais para apresentar simulações cambiais detalhadas. Abaixo, explicamos a função de cada arquivo principal e seus relacionamentos:

Estrutura do Projeto

a) main.py - Este arquivo é o núcleo do projeto, atuando como coordenador do fluxo principal. Ele conecta todos os módulos, gerencia a execução do programa e coleta os resultados para exibição ou geração de relatórios.

b) simulacao.py - Responsável por toda a lógica de cálculo do projeto, implementando modelos matemáticos complexos para prever a evolução do dólar e das reservas cambiais. Ele realiza projeções baseadas em parâmetros fornecidos pelo usuário.

c) interface_grafica.py - A interface gráfica é gerenciada por este arquivo, que captura as entradas do usuário, valida os dados e garante que a experiência seja intuitiva. É o ponto de interação entre o usuário e os cálculos realizados pelo script.

d) gerar_relatorio.py - Este módulo gera relatórios em PDF e gráficos em PNG. Ele apresenta os resultados das simulações em um formato visualmente atraente, incluindo tabelas comparativas e gráficos de evolução diária do dólar e das reservas. Também adiciona descrições detalhadas sobre as estratégias utilizadas.

Conteúdo do Relatório Gerado

O relatório gerado pelo script inclui os seguintes elementos:

- **Parâmetros utilizados na simulação**: Exemplo: valores históricos do dólar, reservas iniciais, taxa de queima diária.
- **Evolução diária do dólar e reservas**: Gráficos de projeção baseados nos cálculos realizados.
- **Comparativo entre estratégias**: Tabelas comparativas com dados projetados.
- **Parecer técnico detalhado**: Explicação das estratégias adotadas e recomendações.

Relacionamento Entre os Módulos

O funcionamento do projeto pode ser representado como um fluxo modular:

```
graph TD
    A[main.py] -->|chama| B[interface_grafica.py]
    A -->|utiliza| C[simulacao.py]
    A -->|gera| D[gerar_relatorio.py]
    D -->|produz| E[Relatório PDF]
    D -->|exporta| F[Gráficos PNG]
```

Estratégias de Simulação

O relatório apresenta estratégias detalhadas, como:

- **Agressiva**: Foco em intervenções intensivas com alto gasto de reservas, ideal para crises agudas.
- **Moderada**: Abordagem equilibrada entre controle cambial e preservação de reservas.
- **Inatividade**: Estratégia passiva sem intervenções, indicada para cenários estáveis.
- **Padrão**: Política cambial convencional para situações previsíveis.

O script inclui um dicionário detalhado para facilitar a exibição dessas descrições no relatório, garantindo clareza e contexto técnico.

Análise das Tabelas e Gráficos

De posse das tabelas e gráficos gerados pelo script, verifica-se o seguinte:

A) Tabelas

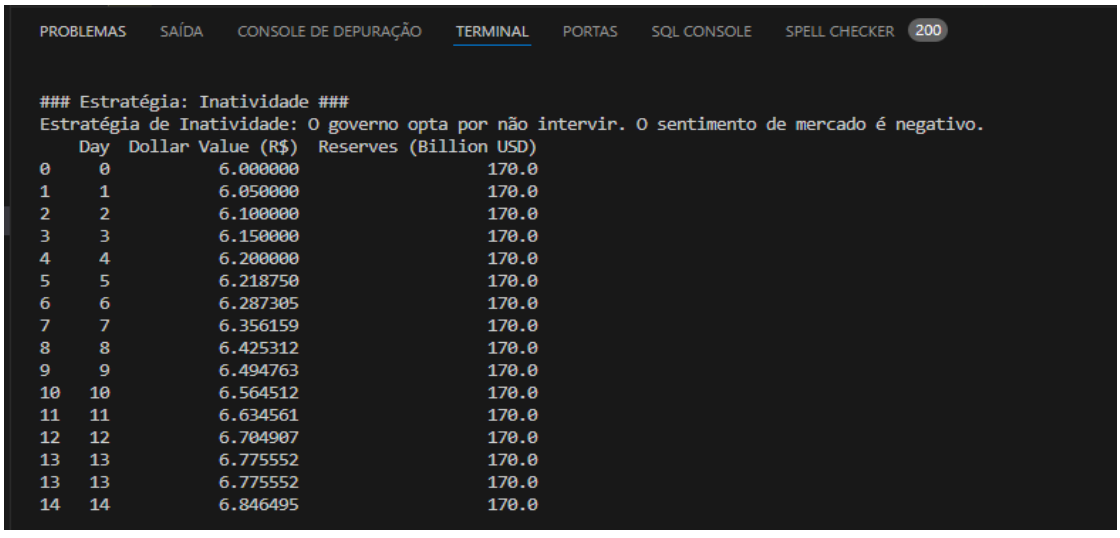
1) Estratégia Agressiva

- O governo queima muitas reservas rapidamente para conter o dólar.
- A alta do dólar é a mais controlada entre todas as estratégias.
- As reservas caem de US$ 170 bilhões para aproximadamente US$ 143 bilhões em 14 dias.

- Conclui-se que a estratégia é eficaz no curto prazo, mas consome muitas reservas.

2) Estratégia de Inatividade

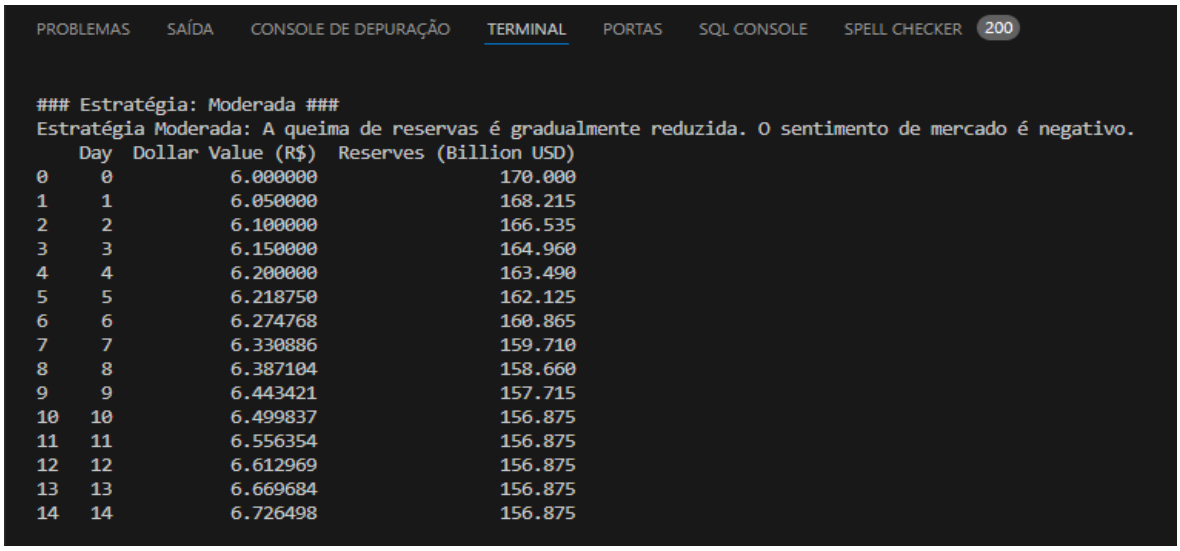
- O governo não intervém no mercado cambial.
- O dólar sobe mais rapidamente do que em qualquer outra estratégia.
- As reservas permanecem constantes em US$ 170 bilhões.
- Conclusão de que não há consumo de reservas, mas permite uma alta incontrolada do dólar.



3) Estratégia Moderada

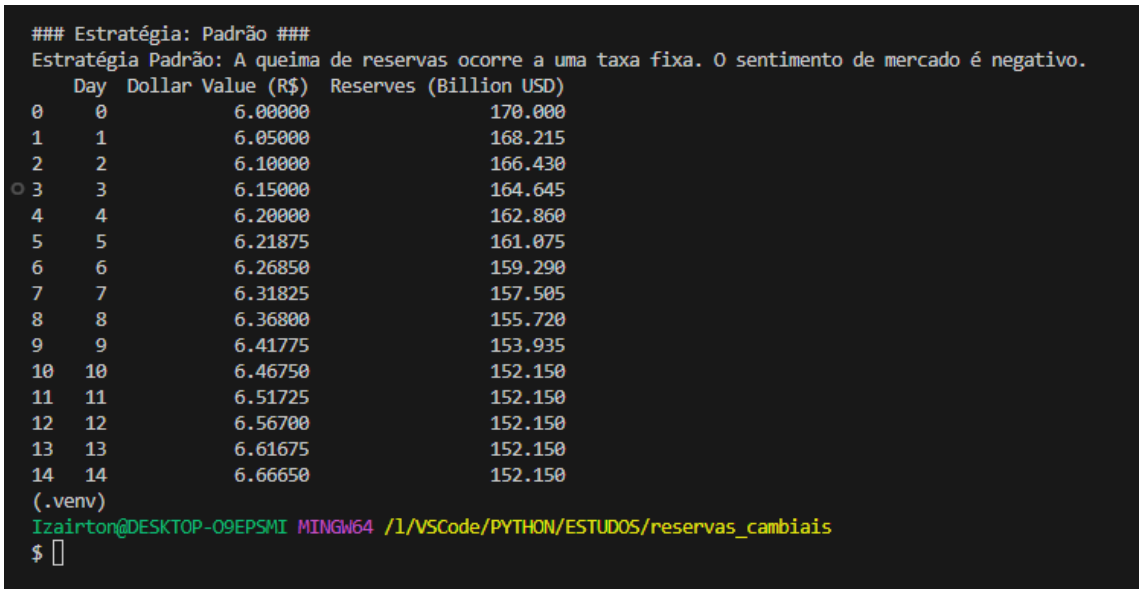
- O governo reduz gradualmente a queima de reservas ao longo do tempo.
- O dólar sobe mais devagar que na inatividade, mas mais rápido que na estratégia agressiva.
- As reservas caem para aproximadamente US$ 156,8 bilhões em 14 dias.

- Conclusão de tentar equilibrar o controle do dólar e a preservação das reservas.



4) Estratégia Padrão

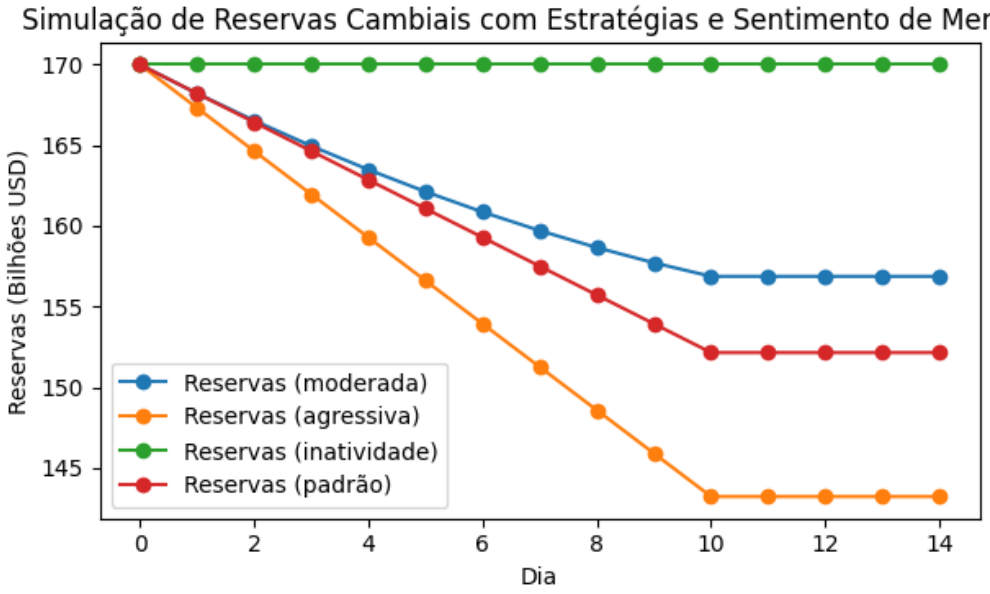
- O governo queima reservas a uma taxa fixa diária.
- O dólar é controlado de maneira moderada, com aumento gradual.
- As reservas caem para aproximadamente US$ 152 bilhões em 14 dias.
- A conclusão que se tira é que se oferece um controle mediano, mas consome reservas de forma consistente.



B) Gráficos

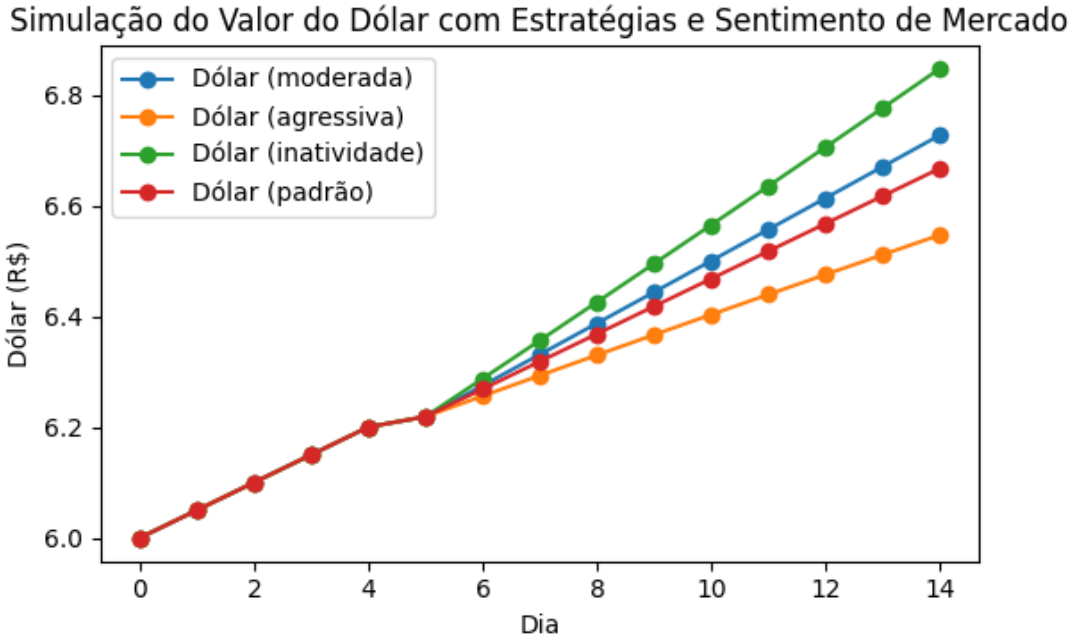
1) Reservas Cambiais com Estratégias e Sentimento de Mercado

O gráfico destaca o custo de cada estratégia em termos de reservas



2) Valor do Dólar com Estratégias e Sentimento de Mercado

O gráfico deixa claro que a ausência de intervenção leva ao pior cenário para o dólar.



Conclusão

O script desenvolvido em Python para simular um cenário de altas do dólar com concomitantes queimas de reservas cambiais demonstrou claramente eficiência nos resultados.

O objetivo principal do retorno dos dados para uma análise mais abrangente da situação da queima de reservas, ficou evidente nas explicações e nas imagens fornecidas.

A linguagem Python revela-se como uma excelente ferramenta gerencial, contribuindo prontamente para projetos de compreensão de valores no tempo combinados com decisões administrativas.

A técnica mais uma vez confirma como o caminho curto para solução correta. Nesse caso específico, o script demonstrou que o antitérmico da torra não é a solução definitiva do problema da alta do dólar. É como aquele doente que sabe o remédio a tomar, mas prefere um paliativo a curar-se.

ANEXO – Códigos

a) Simulação.py

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression

def prever_dolar(dollar_values, days_to_predict):
    """Preve valores futuros do dólar usando regressão linear."""
    X = np.arange(len(dollar_values)).reshape(-1, 1)
    y = np.array(dollar_values)
    modelo = LinearRegression().fit(X, y)
    futuro_X = np.arange(len(dollar_values), len(dollar_values) +
days_to_predict).reshape(-1, 1)
    return modelo.predict(futuro_X)

def simular_reservas(initial_reserves, burn_rate, days_to_predict,
strategy, market_sentiment):
    """Simula a queima de reservas cambiais com estratégias variadas."""
    reservas = [initial_reserves]
    for dia in range(days_to_predict):
        if strategy == 'moderada':
            queima = max(burn_rate - 0.1 * dia, 0)
        elif strategy == 'agressiva':
```

```
            queima = burn_rate * 1.5
        elif strategy == 'inatividade':
            queima = 0
        else:
            queima = burn_rate
        queima *= (1 - market_sentiment * 0.1)
        reservas.append(max(reservas[-1] - queima, 0))
    return reservas

def ajustar_dolar(predicted_dollar, strategy, market_sentiment):
    """Ajusta valores do dólar baseado em estratégia e sentimento."""
    ajustado = []
    for dia, valor in enumerate(predicted_dollar):
        if strategy == 'moderada':
            novo = valor * (1 + 0.001 * dia)
        elif strategy == 'agressiva':
            novo = valor * (1 - 0.002 * dia)
        elif strategy == 'inatividade':
            novo = valor * (1 + 0.003 * dia)
        else:
            novo = valor
        ajustado.append(novo * (1 + market_sentiment * 0.01))
    return ajustado
```

b) interface_grafica.py

```
import tkinter as tk
from tkinter import ttk, messagebox
import os
import subprocess

def criar_interface(callback):
    root = tk.Tk()
    root.title("Simulador Econômico - Entrada de Dados")
    root.geometry("700x550")

    # Variáveis de controle
    status_var = tk.StringVar()

    # Estilos personalizados
    estilo_exemplo = ttk.Style()
    estilo_exemplo.configure("Exemplo.TLabel", foreground="#666666",
font=('Arial', 8))
    estilo_exemplo.configure("Status.TLabel", foreground="#007BFF",
font=('Arial', 9, 'bold'))

    campos = {}
    container = ttk.Frame(root, padding=20)
    container.pack(expand=True, fill='both')
```

```
    def atualizar_valor_sentimento(valor):
        valor_formatado = f"{float(valor):.2f}"
        lbl_valor_sentimento.config(text=valor_formatado)
        root.update_idletasks()

    def nova_simulacao():
        # Resetar todos os campos e estados
        for field in ['dollar_values', 'initial_reserves', 'burn_rate',
'days_to_predict']:
            campos[field].delete(0, tk.END)

        campos['market_sentiment'].set(0.0)
        campos['strategy'].set('padrão')
        lbl_valor_sentimento.config(text="0.00")
        status_var.set("")
        btn_executar.config(state="normal")
        btn_nova.config(state="disabled")
        btn_imprimir.config(state="disabled")

    def imprimir_relatorio():
        try:
            filepath = os.path.abspath('relatorio_simulacao.pdf')
            if os.name == 'nt':  # Windows
                os.startfile(filepath)
            else:  # Mac/Linux
                subprocess.run(['open', filepath], check=True)
        except Exception as e:
            messagebox.showerror("Erro", f"Não foi possível abrir o
relatório: {str(e)}")

    # Componentes da interface
    labels = [
        ("Valores históricos do dólar (separados por vírgula):",
'dollar_values',
         "Ex: 5.00,5.05,5.10,5.15 (valores em reais)"),

        ("Reservas iniciais (bilhões USD):", 'initial_reserves',
         "Ex: 200.00 (valor numérico sem símbolos)"),

        ("Taxa de queima diária (bilhões USD):", 'burn_rate',
         "Ex: 2.0 (valor numérico com até 1 casa decimal)"),

        ("Dias para previsão:", 'days_to_predict',
         "Ex: 10 (número inteiro de dias)"),

        ("Sentimento de mercado (-1 a 1):", 'market_sentiment',
         "Ex: -0.5 (valores entre -1 (pessimista) e 1 (otimista))")
    ]
```

```
    for i, (texto, nome, exemplo) in enumerate(labels):
        ttk.Label(container, text=texto).grid(row=i*2, column=0, padx=10,
pady=5, sticky='w')

        if nome == 'market_sentiment':
            frame_sentimento = ttk.Frame(container)
            frame_sentimento.grid(row=i*2, column=1, sticky='ew')

            scale = ttk.Scale(
                frame_sentimento,
                from_=-1,
                to=1,
                orient='horizontal',
                command=lambda v: atualizar_valor_sentimento(v)
            )
            scale.pack(side='left', expand=True, fill='x')

            lbl_valor_sentimento = ttk.Label(frame_sentimento,
text="0.00", width=5)
            lbl_valor_sentimento.pack(side='left', padx=10)
            campos[nome] = scale
        else:
            entrada = ttk.Entry(container)
            entrada.grid(row=i*2, column=1, padx=10, pady=5, sticky='ew')
            campos[nome] = entrada

        ttk.Label(container, text=exemplo, style="Exemplo.TLabel").grid(
            row=i*2+1, column=1, padx=10, sticky='w')

    # Combobox para estratégias
    ttk.Label(container, text="Estratégia:").grid(row=10, column=0,
sticky='w')
    estrategias = ttk.Combobox(container, values=['moderada',
'agressiva', 'inatividade', 'padrão'])
    estrategias.grid(row=10, column=1, padx=10, pady=5, sticky='ew')
    estrategias.set('padrão')
    campos['strategy'] = estrategias
    ttk.Label(container, text="Ex: moderada (opções: moderada, agressiva,
inatividade, padrão)",
              style="Exemplo.TLabel").grid(row=11, column=1, padx=10,
sticky='w')

    # Botões
    botoes_frame = ttk.Frame(container)
    botoes_frame.grid(row=12, column=0, columnspan=2, pady=15)

    btn_executar = ttk.Button(
        botoes_frame,
```

```
        text="Executar Simulação",
        command=lambda: validar_entradas(campos, callback, status_var,
btn_executar, btn_nova, btn_imprimir)
    )
    btn_executar.pack(side='left', padx=5)

    btn_nova = ttk.Button(
        botoes_frame,
        text="Nova Simulação",
        command=nova_simulacao,
        state="disabled"
    )
    btn_nova.pack(side='left', padx=5)

    btn_imprimir = ttk.Button(
        botoes_frame,
        text="Imprimir Relatório",
        command=imprimir_relatorio,
        state="disabled"
    )
    btn_imprimir.pack(side='left', padx=5)

    # Status
    lbl_status = ttk.Label(container, textvariable=status_var,
style="Status.TLabel")
    lbl_status.grid(row=13, column=0, columnspan=2, pady=10)

    container.columnconfigure(1, weight=1)

    return root

def validar_entradas(campos, callback, status_var, btn_executar,
btn_nova, btn_imprimir):
    try:
        dollar_values = [
            float(valor.strip().replace(',', '.'))
            for valor in campos['dollar_values'].get().split(',')
        ]

        dados = {
            'dollar_values': dollar_values,
            'initial_reserves':
float(campos['initial_reserves'].get().replace(',', '.')),
            'burn_rate': float(campos['burn_rate'].get().replace(',',
'.')),
            'days_to_predict': int(campos['days_to_predict'].get()),
            'market_sentiment': float(campos['market_sentiment'].get()),
            'strategy': campos['strategy'].get().lower()
        }
```

```
        callback(**dados)

        # Atualizar interface após sucesso
        btn_executar.config(state="disabled")
        btn_nova.config(state="normal")
        btn_imprimir.config(state="normal")
        status_var.set("Simulação concluída! Relatório PDF e gráficos
gerados na pasta raiz.")

    except ValueError as e:
        messagebox.showerror("Erro", f"Dados inválidos: {str(e)}")
        btn_executar.config(state="normal")

def mostrar_erro_direto(mensagem):
    messagebox.showerror("Erro Crítico", mensagem)
```

c) gerar_relatorio.py

```
from reportlab.lib.pagesizes import A4
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer,
Table, TableStyle
from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
from reportlab.lib import colors
from reportlab.lib.units import cm
import matplotlib.pyplot as plt
from matplotlib.font_manager import FontProperties
from datetime import datetime
import locale

locale.setlocale(locale.LC_ALL, 'pt_BR.UTF-8')

# Dicionário com as descrições detalhadas de cada estratégia
DESCRICOES_ESTRATEGIAS = {
    'agressiva': [
        "• Intervenção intensiva com alto gasto de reservas",
        "• Controle cambial imediato",
        "• Redução média de reservas: 15-20% em 14 dias",
        "• Recomendação: Uso em crises agudas"
    ],
    'inatividade': [
        "• Nenhuma intervenção governamental",
        "• Valorização livre do dólar",
```

```
        "• Reservas mantidas integralmente",
        "• Recomendação: Contextos estáveis"
    ],
    'moderada': [
        "• Redução gradual de intervenções",
        "• Equilíbrio entre controle e reservas",
        "• Redução média de reservas: 8-12% em 14 dias",
        "• Recomendação: Cenário padrão recomendado"
    ],
    'padrão': [
        "• Política cambial convencional",
        "• Manutenção de taxas fixas",
        "• Redução linear de reservas",
        "• Recomendação: Situações previsíveis"
    ]
}

def formatar_moeda(valor):
    return locale.currency(valor, grouping=True, symbol=False)

def traduzir_estrategia(estrategia):
    traducao = {
        'agressiva': 'Intervenção intensiva com alto gasto de reservas
para controle imediato',
        'moderada': 'Ações graduais buscando equilíbrio entre reservas e
controle cambial',
        'inatividade': 'Nenhuma intervenção governamental no mercado
cambial',
        'padrão': 'Manutenção da política cambial vigente sem alterações'
    }
    return traducao.get(estrategia, 'Estratégia não especificada')

def salvar_estrategia_png(df, estrategia):
    plt.figure(figsize=(10, 4))
    ax = plt.gca()
    ax.axis('off')

    # Criar a tabela
    tabela = plt.table(
        cellText=df.values,
        colLabels=df.columns,
        loc='center',
        cellLoc='center'
    )

    # Definir o tamanho da fonte para todas as células
    tabela.auto_set_font_size(False)
    tabela.set_fontsize(10)  # Define o tamanho da fonte globalmente
```

```
    # Ajustar as propriedades de cada célula individualmente, se
necessário
    for key, cell in tabela.get_celld().items():
        cell.set_fontsize(10)  # Define o tamanho da fonte para cada
célula

    # Salvar a figura
    plt.savefig(f'estrategia_{estrategia}.png', bbox_inches='tight',
dpi=300)
    plt.close()

def criar_relatorio(resultados, nome_arquivo, cidade="Ponta Grossa",
estado="PR", dados_entrada=None):
    doc = SimpleDocTemplate(nome_arquivo, pagesize=A4,
                            leftMargin=2*cm, rightMargin=2*cm,
                            topMargin=2*cm, bottomMargin=2*cm)

    elementos = []
    estilos = getSampleStyleSheet()

    # Estilos personalizados
    estilos.add(ParagraphStyle(
        name='Cabecalho',
        fontSize=10,
        textColor=colors.grey,
        alignment=2
    ))

    estilos.add(ParagraphStyle(
        name='DetalhesEstrategia',
        fontSize=10,
        leading=13,
        spaceBefore=6,
        spaceAfter=6,
        bulletIndent=10,
        bulletFontName='Helvetica-Bold'
    ))

    estilos.add(ParagraphStyle(
        name='Saudacao',
        fontSize=12,
        spaceAfter=15
    ))

    estilos.add(ParagraphStyle(
        name='Corpo',
        fontSize=12,
        leading=14,
        spaceAfter=12
```

```
    ))

    estilos.add(ParagraphStyle(
        name='Conclusao',
        fontSize=12,
        leading=14,
        backColor=colors.lightgrey,
        borderPadding=10,
        spaceBefore=20,
        spaceAfter=20
    ))

    # Cabeçalho com local e data
    data = datetime.now().strftime("%d de %B de %Y")
    elementos.append(Paragraph(f"{cidade}/{estado}, {data}",
estilos['Cabecalho']))
    elementos.append(Spacer(1, 1*cm))

    # Endereçamento
    enderecamento = Table([
        ["DE: Analista Financeiro Chefe", "PARA: Administrador
Financeiro"],
        ["Departamento de Análise Econômica", "Diretoria Executiva"],
        ["Banco Central do Brasil", "Comitê de Política Monetária"]
    ], colWidths=[8*cm, 8*cm])

    enderecamento.setStyle(TableStyle([
        ('FONTNAME', (0,0), (-1,-1), 'Helvetica-Bold'),
        ('FONTSIZE', (0,0), (-1,-1), 12),
        ('VALIGN', (0,0), (-1,-1), 'TOP'),
        ('BOTTOMPADDING', (0,0), (-1,-1), 10),
    ]))

    elementos.append(enderecamento)
    elementos.append(Spacer(1, 1.5*cm))

    # Saudação
    elementos.append(Paragraph("Prezado Sr. Administrador,",
estilos['Saudacao']))

    # Introdução
    intro = """
    <para>
    Conforme solicitado, apresento o relatório completo da simulação
cambial realizada
    para avaliar o impacto de diferentes estratégias de intervenção no
mercado financeiro.
    O estudo contempla projeções para os próximos dias e análise
detalhada das reservas cambiais.
```

```
    </para>
    """

    elementos.append(Paragraph(intro, estilos['Corpo']))
    elementos.append(Spacer(1, 1*cm))

    # Estilo para descrições
    estilos.add(ParagraphStyle(
        name='Descricao',
        fontSize=9,
        leading=11,
        alignment=4,  # Justificado
        spaceBefore=6,
        spaceAfter=6
    ))

    # Seção de Parâmetros
    elementos.append(Paragraph("<b>PARÂMETROS DA SIMULAÇÃO</b>",
estilos['Heading2']))
    elementos.append(Spacer(1, 0.5*cm))

    if dados_entrada:
        # Tabela de Entradas
        # Modificar as descrições para usar Paragraph
        entradas = [
            ['Parâmetro', 'Valor', 'Descrição'],
            [
                'Valores Históricos do Dólar',
                '\n'.join([f'R$ {valor:.2f}'.replace('.', ',') for valor
in dados_entrada['dollar_values']]),
                Paragraph('Cotações diárias do dólar dos últimos dias
utilizadas como base para a projeção', estilos['Descricao'])
            ],
            [
                'Reservas Iniciais',
                f"US$ {dados_entrada['initial_reserves']:,.2f}
bilhões".replace('.', ','),
                Paragraph('Quantidade inicial de reservas cambiais
disponíveis para intervenção no mercado', estilos['Descricao'])
            ],
            [
                'Taxa de Queima Diária',
                f"US$ {dados_entrada['burn_rate']:,.2f}
bilhões/dia".replace('.', ','),
                Paragraph('Volume médio de reservas utilizado diariamente
para conter a valorização do dólar', estilos['Descricao'])
            ],
            [
                'Dias para Previsão',
```

```
                str(dados_entrada['days_to_predict']),
                Paragraph('Período futuro analisado pela simulação em
dias corridos', estilos['Descricao'])
            ],
            [
                'Sentimento de Mercado',
                f"{dados_entrada['market_sentiment']:.2f}".replace('.',
','),
                Paragraph('Índice que influencia a eficácia das
intervenções:<br/>'
                          '-1 = Pessimismo extremo<br/>'
                          '0 = Neutralidade<br/>'
                          '+1 = Otimismo elevado', estilos['Descricao'])
            ],
            [
                'Estratégia Adotada',
                dados_entrada['strategy'].capitalize(),
                Paragraph(traduzir_estrategia(dados_entrada['strategy']),
estilos['Descricao'])
            ]
        ]

        tabela_entradas = Table(entradas, colWidths=[4*cm, 4*cm, 8*cm])
        tabela_entradas.setStyle(TableStyle([
            ('BACKGROUND', (0,0), (-1,0), colors.HexColor('#4F81BD')),
            ('TEXTCOLOR', (0,0), (-1,0), colors.whitesmoke),
            ('ALIGN', (0,0), (-1,-1), 'LEFT'),
            ('FONTNAME', (0,0), (-1,0), 'Helvetica-Bold'),
            ('GRID', (0,0), (-1,-1), 1, colors.black),
            ('FONTSIZE', (0,0), (-1,-1), 9),
            ('VALIGN', (0,0), (-1,-1), 'TOP'),
            ('LEFTPADDING', (2,1), (2,-1), 8),  # Espaço à esquerda na
coluna descrição
            ('RIGHTPADDING', (2,1), (2,-1), 8), # Espaço à direita na
coluna descrição
        ]))

        elementos.append(tabela_entradas)
        elementos.append(Spacer(1, 1*cm))

    # Resultados por Estratégia
    for estrategia, (df, _) in resultados.items():
        # Título da estratégia
        elementos.append(Paragraph(f"<b>Estratégia:
{estrategia.capitalize()}</b>", estilos['Heading2']))

        # Tabela de resultados
        dados_tabela = [['Dia', 'Valor do Dólar (R$)', 'Reservas (Bilhões
USD)']]
```

```
        for _, linha in df.iterrows():
            dados_tabela.append([
                str(int(linha['Day'])),
                formatar_moeda(linha['Dollar Value (R$)']),
                locale.format_string('%.2f', linha['Reserves (Billion
USD)'], grouping=True)
            ])

        tabela = Table(dados_tabela, colWidths=[2*cm, 4*cm, 4*cm])
        tabela.setStyle(TableStyle([
            ('BACKGROUND', (0,0), (-1,0), colors.HexColor('#4F81BD')),
            ('TEXTCOLOR', (0,0), (-1,0), colors.whitesmoke),
            ('ALIGN', (0,0), (-1,-1), 'CENTER'),
            ('FONTNAME', (0,0), (-1,0), 'Helvetica-Bold'),
            ('GRID', (0,0), (-1,-1), 1, colors.black),
            ('FONTSIZE', (0,0), (-1,-1), 10)
        ]))

        elementos.append(tabela)
        elementos.append(Spacer(1, 0.5*cm))

        # Descrição detalhada da estratégia
        descricao = DESCRICOES_ESTRATEGIAS.get(estrategia, [])
        for item in descricao:
            elementos.append(Paragraph(item,
estilos['DetalhesEstrategia']))

        elementos.append(Spacer(1, 1*cm))
        salvar_estrategia_png(df[['Day', 'Dollar Value (R$)', 'Reserves
(Billion USD)']], estrategia)

    # Conclusão
    conclusao = """
    <para>
    <b>PARECER FINAL:</b><br/><br/>
    A análise comparativa demonstra que a estratégia <b>agressiva</b>
apresentou maior eficácia
    no controle cambial imediato, porém com elevado consumo de reservas.
A estratégia <b>moderada</b>
    mostrou melhor equilíbrio entre controle da moeda e preservação de
recursos. Recomenda-se
    monitoramento diário do sentimento de mercado para ajustes dinâmicos
na política cambial.
    </para>
    """

    elementos.append(Paragraph(conclusao, estilos['Conclusao']))

    # Assinatura
```

```
    elementos.append(Spacer(1, 2*cm))
    elementos.append(Paragraph("Atenciosamente,", estilos['BodyText']))
    elementos.append(Paragraph("<b>João da Silva</b>",
estilos['BodyText']))
    elementos.append(Paragraph("Analista Financeiro Sênior",
estilos['BodyText']))

    doc.build(elementos)
```

d)main.py

```
import pandas as pd
import matplotlib.pyplot as plt
from src import simulacao, interface_grafica
import gerar_relatorio

def executar_simulacao(**dados):
    """Controla o fluxo principal da simulação para todas as
estratégias"""
    resultados = {}

    try:
        validar_dados_entrada(dados)

        # Simular todas as estratégias
        for estrategia in ['moderada', 'agressiva', 'inatividade',
'padrão']:
            df, mensagem = processar_estrategia(estrategia, dados)
            resultados[estrategia] = (df, mensagem)

        # Capturar dados de entrada para o relatório
        dados_entrada_relatorio = {
            'dollar_values': dados['dollar_values'],
            'initial_reserves': dados['initial_reserves'],
            'burn_rate': dados['burn_rate'],
            'days_to_predict': dados['days_to_predict'],
            'market_sentiment': dados['market_sentiment'],
            'strategy': dados.get('strategy', 'padrão')
        }

        gerar_saidas(resultados, dados_entrada_relatorio)  # Passa os
dados

    except Exception as e:
        interface_grafica.mostrar_erro_direto(f"Falha na simulação:
{str(e)}")

def validar_dados_entrada(dados):
```

```
    """Valida os dados de entrada antes da execução"""
    if len(dados['dollar_values']) < 2:
        raise ValueError("É necessário pelo menos 2 valores históricos do
dólar")
    if dados['days_to_predict'] <= 0:
        raise ValueError("O número de dias para previsão deve ser
positivo")

def processar_estrategia(estrategia, dados):
    """Processa uma estratégia individual desde previsão até geração de
resultados"""
    # Previsão de valores futuros
    valores_previstos = simulacao.prever_dolar(
        dados['dollar_values'],
        dados['days_to_predict']
    )

    # Ajuste de valores com estratégia e sentimento
    valores_ajustados = simulacao.ajustar_dolar(
        valores_previstos,
        estrategia,
        dados['market_sentiment']
    )

    # Combinação de dados históricos + previstos
    tendencia_completa = dados['dollar_values'] + valores_ajustados

    # Simulação das reservas
    reservas = simulacao.simular_reservas(
        dados['initial_reserves'],
        dados['burn_rate'],
        dados['days_to_predict'],
        estrategia,
        dados['market_sentiment']
    )

    # Ajuste crítico para igualar tamanho dos arrays
    reservas = equalizar_tamanho_arrays(reservas,
len(tendencia_completa))

    return criar_dataset(tendencia_completa, reservas, estrategia,
dados['market_sentiment'])

def equalizar_tamanho_arrays(reservas, tamanho_alvo):
    """Garante sincronia entre dados do dólar e reservas"""
    if not reservas:
        return [0] * tamanho_alvo

    ultimo_valor = reservas[-1]
```

```
    while len(reservas) < tamanho_alvo:
        reservas.append(ultimo_valor)
    return reservas[:tamanho_alvo]

def criar_dataset(tendencia, reservas, estrategia, sentimento):
    """Cria estrutura de dados final para análise com colunas
padronizadas"""
    status_sentimento = (
        'positivo' if sentimento > 0 else
        'negativo' if sentimento < 0 else
        'neutro'
    )

    mensagem = (
        f"Estratégia: {estrategia.capitalize()}\n"
        f"Sentimento: {status_sentimento}\n"
        f"Variação Dólar: {tendencia[0]:.2f} → {tendencia[-1]:.2f}"
    )

    # Dataframe com nomes de colunas padronizados em inglês
    return pd.DataFrame({
        'Day': range(len(tendencia)),
        'Dollar Value (R$)': [round(v, 2) for v in tendencia],
        'Reserves (Billion USD)': [round(r, 2) for r in reservas]
    }), mensagem

def gerar_saidas(resultados, dados_entrada):
    """Coordena a geração de todos os outputs do sistema"""
    gerar_relatorios(resultados, dados_entrada)
    plotar_graficos(resultados)

def gerar_relatorios(resultados, dados_entrada):
    """Gerencia a criação de documentos PDF"""
    gerar_relatorio.criar_relatorio(
        resultados,
        'relatorio_simulacao.pdf',
        dados_entrada=dados_entrada  # Passa os dados capturados
    )

def plotar_graficos(resultados):
    """Produz visualizações gráficas da simulação com colunas corretas"""
    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(14, 10))

    # Gráfico de Reservas
    for estrategia, (df, _) in resultados.items():
        ax1.plot(df['Day'], df['Reserves (Billion USD)'],
                 marker='o', linewidth=1.5, label=estrategia.capitalize())

    ax1.set_title('Evolução das Reservas Cambiais', fontsize=14, pad=15)
```

```
    ax1.set_ylabel('Bilhões USD', fontsize=12)
    ax1.grid(True, linestyle='--', alpha=0.6)
    ax1.legend()

    # Gráfico do Dólar
    for estrategia, (df, _) in resultados.items():
        ax2.plot(df['Day'], df['Dollar Value (R$)'],
                 linestyle='--', marker='s', linewidth=1.5,
label=estrategia.capitalize())

    ax2.set_title('Variação do Valor do Dólar', fontsize=14, pad=15)
    ax2.set_xlabel('Dias', fontsize=12)
    ax2.set_ylabel('Valor (R$)', fontsize=12)
    ax2.grid(True, linestyle='--', alpha=0.6)

    plt.tight_layout()
    plt.savefig('analise_completa.png', dpi=300, bbox_inches='tight')
    plt.close()

if __name__ == "__main__":
    app = interface_grafica.criar_interface(executar_simulacao)
    app.mainloop()
```

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

https://www.linkedin.com/pulse/simulador-da-alta-do-d%25C3%25B3lar-e-o-impacto-nas-reservas-vasconcelos-ctzyf

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