Taller_1

March 6, 2026

1 Taller 1 — Macroeconomía 3: Ciclos Económicos

1.1 Universidad EAFIT

Estudiantes: Santiago Tupaz - Emmanuel Piedrahita **Profesor:** Álvaro Arturo Hurtado
Monitor: Diego Alejandro Quintero
Fecha: Marzo 6, 2026

1.2 Descripción del documento

Este notebook presenta el desarrollo completo del **Taller 1** del curso *Macroeconomía 3: Ciclos Económicos*.
El trabajo está organizado en **fases**, cada una con un objetivo claro, procedimientos replicables y una interpretación económica concisa.

1.3 Estructura por fases

1. **Fase 1 — Datos y exploración descriptiva:** construcción del dataset, estadísticas descriptivas, visualización y periodos con contracción del PIB.

2. **Fase 2 — Filtro Hodrick–Prescott desde cero:** implementación manual del filtro, descomposición tendencia–ciclo y brechas.

3. **Fase 3 — HP con librería y hechos estilizados:** brechas para todas las variables, volatilidad (absoluta y relativa) y correlaciones (contemporáneas y móviles).

4. **Fase 4 — Matriz insumo–producto:** eslabonamientos directos e indirectos (hacia atrás y hacia adelante) e interpretación económica.

5. **Fase 5 — Empaquetado final:** exportación del Excel final con resultados consolidados y generación del PDF/HTML de entrega.

2 1. Fase 1 — Datos y hechos preliminares

Objetivo: Este notebook documenta el desarrollo del Taller 1 para Macroeconomía 3 (Ciclos Económicos). En la Fase 1 cargamos y exploramos datos macroeconómicos de EE. UU. (FRED), generamos estadísticas descriptivas, visualizamos 7 series trimestrales e identificamos períodos con

crecimiento negativo del PIB real. En las siguientes secciones se desarrollan las fases posteriores (HP, hechos estilizados e insumo-producto).

2.1 1.1 Paths & Folders (Detección automática y creación)

[1]:
```
from pathlib import Path
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')

# Detect ROOT by searching upwards for 'data' folder
current = Path.cwd()
ROOT = None

for parent in current.parents:
    if (parent / 'data').exists():
        ROOT = parent
        break

if ROOT is None:
    if (current / 'data').exists():
        ROOT = current
    else:
        raise FileNotFoundError("Could not find 'data' folder. Please run from␣
 ↪project root.")

# Define paths
DATA_RAW = ROOT / 'data' / 'raw'
DATA_PROC = ROOT / 'data' / 'processed'
OUT_FIG = ROOT / 'outputs' / 'figures'
OUT_TAB = ROOT / 'outputs' / 'tables'
OUT_EXP = ROOT / 'outputs' / 'exports'

# Create directories if they don't exist
OUT_FIG.mkdir(parents=True, exist_ok=True)
OUT_TAB.mkdir(parents=True, exist_ok=True)
OUT_EXP.mkdir(parents=True, exist_ok=True)

print("Directorios verificados y listos para outputs.")
```

```
Directorios verificados y listos para outputs.
```

2.2 1.2 Decisiones Económicas: País, Variables y Transformaciones

2.2.1 1.2.1 País y Fuente de Datos

Estados Unidos (FRED) con frecuencia objetivo **trimestral**.

2.2.2 1.2.2 Motivación

1. **Disponibilidad y calidad:** El Federal Reserve Economic Data (FRED) ofrece series de alta calidad y consistencia temporal.
2. **Evaluación del filtro HP:** Utilizamos un país desarrollado para validar cómo se ajusta el filtro Hodrick-Prescott en una economía madura.
3. **Comparación conceptual:** Exploraremos si las dinámicas económicas estadounidenses difieren significativamente de las observadas en Colombia, permitiendo futuros análisis comparativos.

2.2.3 1.2.3 Variables Elegidas

- **GDP (Producto Interno Bruto real):** Medida de actividad económica agregada.
- **Consumo (PCECC96):** Gasto en consumo personal en dólares encadenados 2012.
- **Inversión (GPDIC1):** Inversión fija privada doméstica en dólares encadenados 2012.
- **Desempleo (UNRATE):** Tasa de desempleo (porcentaje).
- **Inflación (GDPDEF):** Deflactor del PIB, refleja cambios de precios en la economía.
- **Tasa de política (FEDFUNDS):** Tasa de fondos federales, herramienta de política monetaria de la Fed.

2.2.4 1.2.4 Propósito

1. **Corto plazo:** Estudiar brechas de componentes del PIB (tendencia vs. ciclo) para entender dinámicas de consumo e inversión.
2. **Extensión futura:** Explorar relaciones empíricas entre brecha del PIB, desempleo e inflación (curva de Phillips, ley de Okun).

2.2.5 1.2.5 Transformaciones Aplicadas

- **UNRATE y FEDFUNDS (mensual → trimestral):** Promedio simple de tres observaciones mensuales por trimestre.
- **GDP Var (tasa de crecimiento anualizada):** $\text{GDP Var} = 400 \times \Delta \log(\text{GDP})$
- **Inflación (tasa anualizada):** $\text{inflation} = 400 \times \Delta \log(\text{GDPDEF})$

2.3 1.3 Cargar Dataset Limpio

[2]:
```
# Load dataset from processed folder
xlsx_file = DATA_PROC / 'dataset_taller.xlsx'
csv_file = DATA_PROC / 'dataset_taller.csv'

if xlsx_file.exists():
    df = pd.read_excel(xlsx_file)
elif csv_file.exists():
    df = pd.read_csv(csv_file)
else:
    raise FileNotFoundError(f"Neither {xlsx_file} nor {csv_file} found.")

# Ensure 'date' column is datetime
df['date'] = pd.to_datetime(df['date'])
```

```
df = df.sort_values('date').reset_index(drop=True)

print(f"Dataset: {df.shape[0]} filas × {df.shape[1]} columnas | Periodo:␣
  ↪{df['date'].min().date()} a {df['date'].max().date()}\n")
display(df.head())
```

```
Dataset: 285 filas × 8 columnas | Periodo: 1954-12-31 a 2025-12-31
```

	date	GDP	GDP Var	consumption	investment	unemployment	inflation	fedfunds
0	1954-12-31	2936.852	7.752247	1761.703	295.882	5.333333	1.088836	0.986667
1	1955-03-31	3020.746	11.266225	1801.231	325.442	4.733333	1.876424	1.343333
2	1955-06-30	3069.910	6.457770	1835.737	345.235	4.400000	1.634681	1.500000
3	1955-09-30	3111.379	5.367116	1858.411	350.533	4.100000	2.786861	1.940000
4	1955-12-31	3130.068	2.395477	1881.748	358.661	4.233333	3.943755	2.356667

[3]:
```
# Data types
print("Tipos de datos:")
print(df.dtypes)

# Validate no missing values in required columns
required_cols = ['GDP', 'GDP Var', 'consumption', 'investment', 'unemployment',␣
  ↪'inflation', 'fedfunds']
assert df[required_cols].isna().sum().sum() == 0, "Valores faltantes detectados␣
  ↪en columnas requeridas"

print("\nSin valores faltantes en columnas requeridas.")
```

```
Tipos de datos:
date            datetime64[us]
GDP                    float64
GDP Var                float64
consumption            float64
investment             float64
unemployment           float64
inflation              float64
fedfunds               float64
dtype: object

Sin valores faltantes en columnas requeridas.
```

2.4 1.4 Tabla Descriptiva

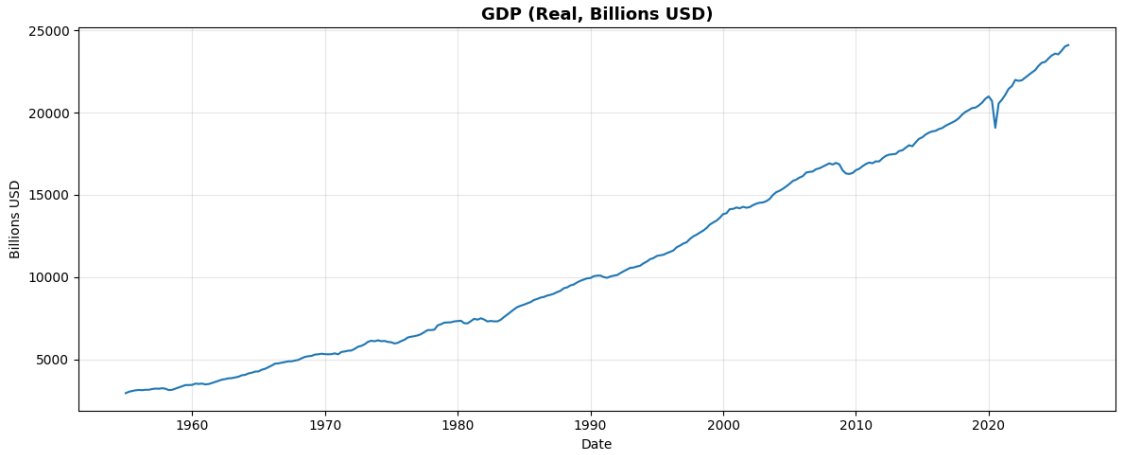
[4]:
```
# Calculate descriptive statistics (exclude date)
desc = df.drop(columns=['date']).describe().round(2)

# Save to Excel and CSV
desc.to_excel(OUT_TAB / 'descriptivas.xlsx')
desc.to_csv(OUT_TAB / 'descriptivas.csv')

# Display table
display(desc)
```

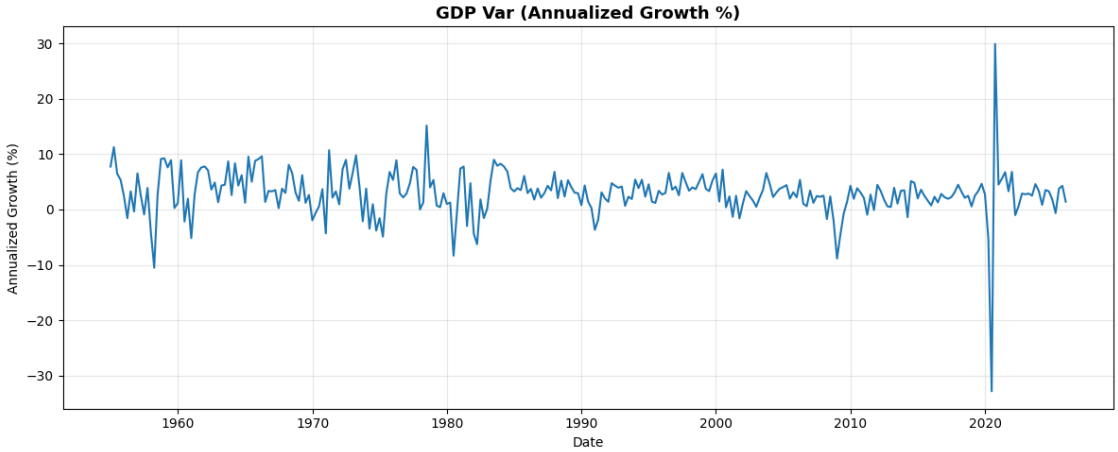
```
            GDP  GDP Var  consumption  investment  unemployment  inflation  \
count    285.00   285.00       285.00      285.00        285.00     285.00
mean   11171.05     2.98      7325.78     1724.60          5.80       3.18
std     6132.25     4.29      4296.59     1206.13          1.67       2.26
min     2936.85   -32.82      1761.70      288.01          3.40      -1.60
25%     5814.85     1.36      3627.90      669.44          4.50       1.66
50%     9998.70     3.07      6353.14     1279.70          5.53       2.48
75%    16561.87     4.65     11158.80     2638.67          6.83       4.03
max    24111.83    29.90     16682.49     4547.95         13.00      11.60

       fedfunds
count    285.00
mean       4.62
std        3.53
min        0.06
25%        1.94
50%        4.33
75%        6.22
max       17.78
```

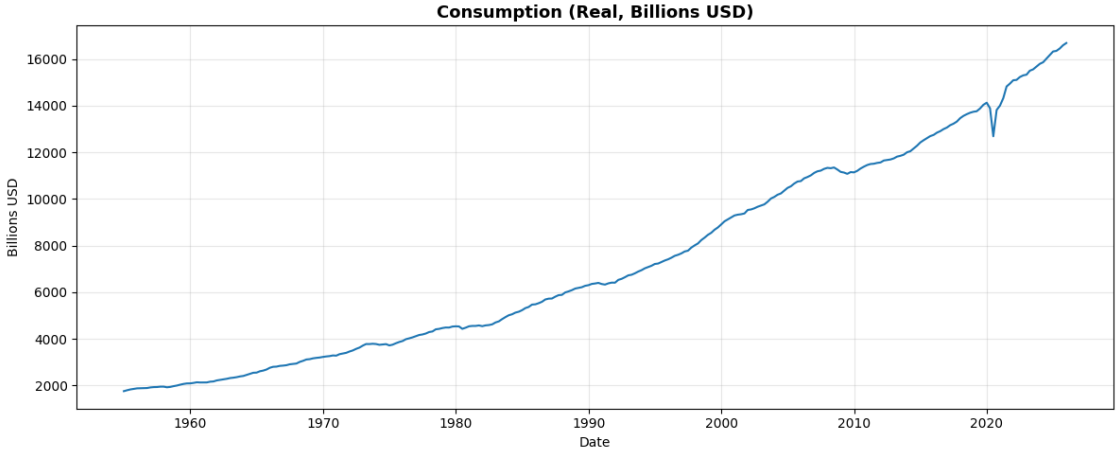
2.5 1.5 Gráficos Individuales por Variable (7 Series)

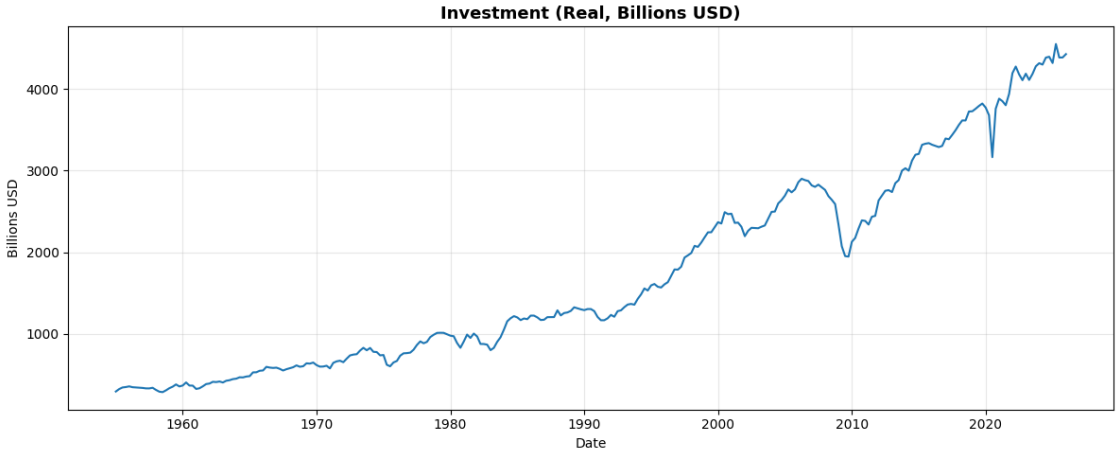
[5]:
```
# Function to create, save and display plots
def plot_and_save(df, col, title, ylabel, filename):
    fig, ax = plt.subplots(figsize=(12, 5))
    ax.plot(df['date'], df[col], linewidth=1.5)
    ax.set_xlabel('Date')
    ax.set_ylabel(ylabel)
    ax.set_title(title, fontsize=13, fontweight='bold')
    ax.grid(True, alpha=0.3)
    plt.tight_layout()

    fig_path = OUT_FIG / filename
    plt.savefig(fig_path, dpi=150, bbox_inches='tight')
    plt.show()
    plt.close()
```

```
# Plot 7 variables
variables = [
    ('GDP', 'GDP (Real, Billions USD)', 'Billions USD', '01_GDP.png'),
    ('GDP Var', 'GDP Var (Annualized Growth %)', 'Annualized Growth (%)',␣
 ↪'02_GDP_Var.png'),
    ('consumption', 'Consumption (Real, Billions USD)', 'Billions USD',␣
 ↪'03_consumption.png'),
    ('investment', 'Investment (Real, Billions USD)', 'Billions USD',␣
 ↪'04_investment.png'),
    ('unemployment', 'Unemployment Rate (%)', 'Unemployment (%)',␣
 ↪'05_unemployment.png'),
    ('inflation', 'Inflation (Annualized %)', 'Annualized (%)', '06_inflation.
 ↪png'),
    ('fedfunds', 'Federal Funds Rate (%)', 'Rate (%)', '07_fedfunds.png')
]

for col, title, ylabel, filename in variables:
    plot_and_save(df, col, title, ylabel, filename)
```

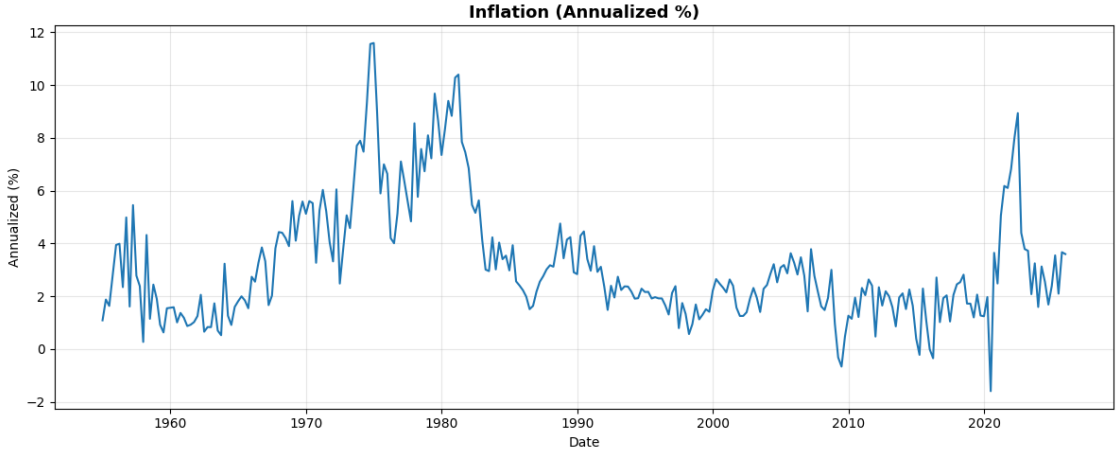


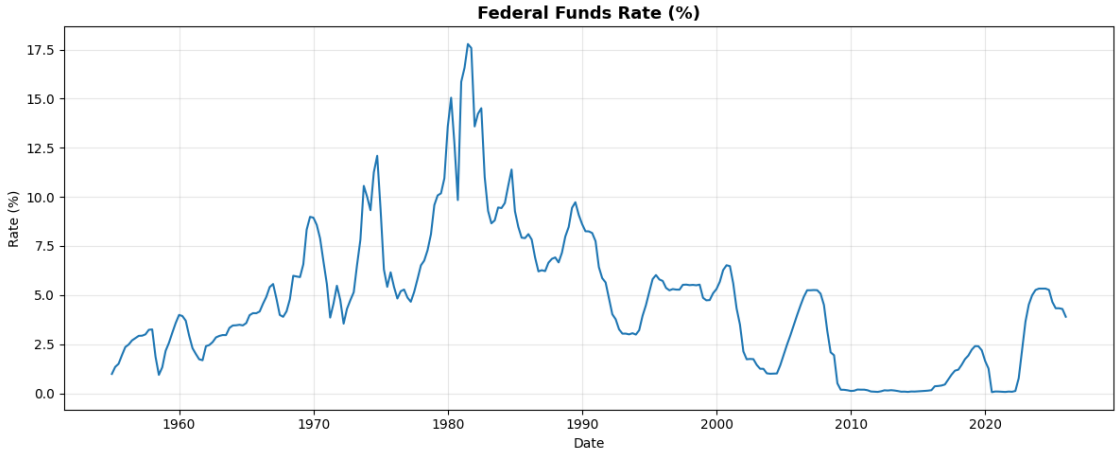

GDP Var (Annualized Growth %)
Annualized Growth (%)
30
20
10
0
−10
−20
−30
1960
1970
1980
1990
2000
2010
2020
Date


Consumption (Real, Billions USD)
Billions USD
16000
14000
12000
10000
8000
6000
4000
2000
1960
1970
1980
1990
2000
2010
2020
Date


Investment (Real, Billions USD)
Billions USD
4000
3000
2000
1000
1960
1970
1980
1990
2000
2010
2020
Date


Unemployment Rate (%)
Unemployment (%)
12
10
8
6
4
1960
1970
1980
1990
2000
2010
2020
Date


Inflation (Annualized %)
Annualized (%)
12
10
8
6
4
2
0
−2
1960
1970
1980
1990
2000
2010
2020
Date


Federal Funds Rate (%)
Rate (%)
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
1960
1970
1980
1990
2000
2010
2020
Date

2.6 1.6 Tabla de Períodos con GDP Var < 0

[6]:
```
# Create recession table
df_neg = df[df['GDP Var'] < 0][['date', 'GDP Var']].copy()

# Prepare for export (convert date to date-only)
df_neg_export = df_neg.copy()
df_neg_export['date'] = df_neg_export['date'].dt.date

# Save to Excel and CSV
df_neg_export.to_excel(OUT_TAB / 'gdp_var_negativa.xlsx', index=False)
df_neg_export.to_csv(OUT_TAB / 'gdp_var_negativa.csv', index=False)

# Display results
print(f"Total de trimestres con GDP Var < 0: {len(df_neg)}\n")
print("Primeros 10 trimestres:")
display(df_neg.head(10))
```

```
Total de trimestres con GDP Var < 0: 37

Primeros 10 trimestres:
```

	date	GDP Var
5	1956-03-31	-1.555190
7	1956-09-30	-0.358975
10	1957-06-30	-0.877523
12	1957-12-31	-4.159708
13	1958-03-31	-10.520768
22	1960-06-30	-2.159247
24	1960-12-31	-5.165738
60	1969-12-31	-1.957619
61	1970-03-31	-0.596011
64	1970-12-31	-4.309423

2.6.1 1.6.1 Análisis de Volatilidad Extrema en GDP Var

[7]:
```
# Find periods of maximum and minimum GDP Var
max_idx = df['GDP Var'].idxmax()
min_idx = df['GDP Var'].idxmin()

max_val = df.loc[max_idx, 'GDP Var']
min_val = df.loc[min_idx, 'GDP Var']
max_date = df.loc[max_idx, 'date']
min_date = df.loc[min_idx, 'date']

# Create summary table
extremes = pd.DataFrame({
    'Período': ['Máxima Expansión', 'Máxima Contracción'],
    'Fecha': [max_date.strftime('%Y-%m-%d'), min_date.strftime('%Y-%m-%d')],
```

```
    'GDP Var (%)': [max_val, min_val]
})

print("VOLATILIDAD EXTREMA EN GDP VAR")
print("=" * 60)
print(f"Máxima Expansión:   {max_date.strftime('%Y-%m-%d')} | GDP Var: {max_val:
  ↪.2f}%")
print(f"Máxima Contracción: {min_date.strftime('%Y-%m-%d')} | GDP Var: {min_val:
  ↪.2f}%")
print(f"Amplitud Total:     {max_val - min_val:.2f} puntos porcentuales")
print("=" * 60)
print()

display(extremes)
```

```
VOLATILIDAD EXTREMA EN GDP VAR
============================================================
Máxima Expansión:   2020-09-30 | GDP Var: 29.90%
Máxima Contracción: 2020-06-30 | GDP Var: -32.82%
Amplitud Total:     62.72 puntos porcentuales
============================================================
```

	Período	Fecha	GDP Var (%)
0	Máxima Expansión	2020-09-30	29.902999
1	Máxima Contracción	2020-06-30	-32.817282

2.7 1.7 Interpretación descriptiva (Punto 1)

2.7.1 1.7.1 Cobertura y consistencia de datos

La muestra cubre el periodo **1954Q4–2025Q4** (frecuencia trimestral). Las series de nivel (**GDP real, consumo real e inversión real**) muestran una tendencia creciente de largo plazo, mientras que las series de tasa (**desempleo, inflación y tasa de política**) exhiben cambios de régimen y episodios de alta volatilidad.

2.7.2 1.7.2 Volatilidad relativa

En niveles, **GDP, consumption e investment** están dominadas por crecimiento secular; por eso, visualmente los ciclos aparecen como desviaciones relativamente pequeñas alrededor de una tendencia. En contraste, **GDP Var** (crecimiento trimestral anualizado) es altamente volátil: al diferenciar ($\Delta\log$) se elimina gran parte de la tendencia y se amplifica la variación de corto plazo.

Comparando **consumo vs inversión**, la inversión muestra fluctuaciones más pronunciadas: cae más en episodios recesivos y se recupera con mayor amplitud. Esto es coherente con hechos estilizados de ciclos: la inversión suele ser el componente más sensible a shocks y condiciones financieras, mientras el consumo es más "suave".

2.7.3 1.7.3 Episodios macroeconómicos identificables en las gráficas

(i) COVID-19 (2020): shock extremo y reversión rápida
El periodo de pandemia concentra los extremos principales de varias series: - **Mayor contracción:** 2020-06-30 con **GDP Var = -32.82%** (anualizado). - **Mayor expansión (rebote):** 2020-09-30 con **GDP Var = 29.90%**. - **Inflación mínima (deflación):** 2020-06-30 con **inflation = -1.60%**. - **Desempleo máximo:** 2020-06-30 con **unemployment = 13.0%**. - **Tasa de política mínima:** 2020-06-30 con **fedfunds = 0.06%**.

Este patrón (colapso + rebote inmediato) es consistente con un shock transitorio muy fuerte (cierres) seguido por reapertura y respuesta macroeconómica expansiva.

(ii) Régimen de alta inflación (1970s)
La inflación anualizada alcanza su máximo en **1974-12-31** con **inflation = 11.60%**, consistente con el episodio de alta inflación asociado a shocks de oferta y tensiones macro del periodo.

(iii) Política monetaria extremadamente contractiva (inicio de 1980s)
La tasa Fed Funds alcanza su máximo en **1981-06-30** con **fedfunds = 17.78%**, coherente con el endurecimiento monetario del inicio de los 80s para reducir la inflación ("Volcker disinflation").

2.7.4 1.7.4 Coherencia macroeconómica entre actividad y desempleo

La relación cualitativa entre actividad y desempleo es coherente con la intuición de ciclos: cuando el crecimiento cae fuertemente (**GDP Var** negativa), el desempleo aumenta (posiblemente con rezago). Esto se observa de forma clara en 2020-06-30, donde el mínimo de GDP Var coincide con el máximo de unemployment.

2.7.5 1.7.5 Periodos de contracción: `GDP Var < 0`

Al filtrar trimestres con **GDP Var < 0**, se identifican **37 trimestres** de contracción en la muestra. Importante: esto no equivale automáticamente a "37 recesiones", ya que una recesión es un episodio que suele agrupar varios trimestres consecutivos. Aun así, esta tabla sirve como indicador simple para ubicar periodos de desaceleración/contracción.

2.7.6 1.7.6 Preparación para Fase 2 (Filtro HP)

Los candidatos naturales para separar tendencia y ciclo son las series en niveles con tendencia fuerte: **GDP, consumption e investment** (idealmente en logaritmos). En contraste, **GDP Var** e **inflation** ya son tasas de cambio y **unemployment** y **fedfunds** son tasas; aplicar HP allí puede ser menos informativo si no se justifica cuidadosamente.

2.8 1.8 Verificación de Outputs Generados

[8]:
```
# List generated outputs with relative paths
print("ARCHIVOS GENERADOS")
print("=" * 60)

print("\nTablas (outputs/tables/):")
for f in sorted(OUT_TAB.glob('*')):
    if f.is_file():
        print(f"  • {f.name}")
```

```python
print("\nGráficos (outputs/figures/):")
for f in sorted(OUT_FIG.glob('*.png')):
    print(f"  • {f.name}")

print("\n Notebook completado exitosamente.")
```

```
ARCHIVOS GENERADOS
============================================================

Tablas (outputs/tables/):
  • correlaciones_hp.csv
  • correlaciones_hp.xlsx
  • correlaciones_moviles_hp.csv
  • correlaciones_moviles_hp.xlsx
  • descriptivas.csv
  • descriptivas.xlsx
  • gdp_var_negativa.csv
  • gdp_var_negativa.xlsx
  • hp_gaps_statsmodels.csv
  • hp_gaps_statsmodels.xlsx
  • io_linkages.xlsx
  • volatilidad_hp.csv
  • volatilidad_hp.xlsx

Gráficos (outputs/figures/):
  • 01_GDP.png
  • 02_GDP_Var.png
  • 03_consumption.png
  • 04_investment.png
  • 05_unemployment.png
  • 06_inflation.png
  • 07_fedfunds.png
  • 08_hp_gdp_trend.png
  • 09_hp_inv_trend.png
  • 10_hp_gdp_cycle.png
  • 11_hp_inv_cycle.png
  • 12_hp_gaps_gdp_inv.png
  • 13_hp_gaps_statsmodels.png
  • 14_rolling_corr.png
  • 15_io_scatter_direct.png

 Notebook completado exitosamente.
```

3 2 Fase 2 — Filtro Hodrick–Prescott (HP) desde cero

El **filtro Hodrick-Prescott** es un método de descomposición tendencia-ciclo ampliamente usado en macroeconomía. Su objetivo es separar una serie temporal en dos componentes: la **tendencia ( )**, que captura el crecimiento suave de largo plazo, y el **ciclo (c)**, que representa desviaciones transitorias alrededor de la tendencia.

Matemáticamente, minimiza:

$$\sum_{t=1}^{T}(y_t - \tau_t)^2 + \lambda \sum_{t=2}^{T-1}(\Delta^2 \tau_t)^2$$

El parámetro **(lambda)** controla el equilibrio entre ajuste (primer término) y suavidad de la tendencia (segundo término). Para datos **trimestrales**, la convención estándar es **= 1600**, que penaliza cambios en la aceleración de la tendencia lo suficiente para generar una descomposición económicamente interpretable.

En esta fase, implementamos el filtro manualmente usando álgebra lineal (desde `src/hp_filter.py`) y lo aplicamos a **GDP** e **Investment** para obtener sus componentes tendencia-ciclo y brechas de producto.

3.1 2.1 Imports y Datos (Reutilizando Fase 1)

[9]:
```
import sys

# Ensure src is in path for imports from hp_filter
src_path = ROOT / 'src'
if str(src_path) not in sys.path:
    sys.path.append(str(src_path))

from hp_filter import hp_manual, gap_ratio

# Verify data is available from Phase 1
print(f"Dataset cargado: {df.shape[0]} observaciones × {df.shape[1]} variables")
print(f"Columnas disponibles: {list(df.columns)}")
print("  Listos para descomposición HP.")
```

```
Dataset cargado: 285 observaciones × 8 variables
Columnas disponibles: ['date', 'GDP', 'GDP Var', 'consumption', 'investment',
'unemployment', 'inflation', 'fedfunds']
  Listos para descomposición HP.
```

3.2 2.2 Preparar Series (Log Niveles)

[10]:
```
# Prepare series in log levels for HP decomposition
y_gdp = np.log(df["GDP"].to_numpy())
y_inv = np.log(df["investment"].to_numpy())

# HP smoothness parameter for quarterly data
lam = 1600
```

```
print(f"Series en log (GDP): {y_gdp.shape}")
print(f"Series en log (Investment): {y_inv.shape}")
print(f"Parámetro  (lambda): {lam}")
print("  Series preparadas para filtro HP.")
```

```
Series en log (GDP): (285,)
Series en log (Investment): (285,)
Parámetro  (lambda): 1600
  Series preparadas para filtro HP.
```

3.3 2.3 Aplicar HP Manual (Log)

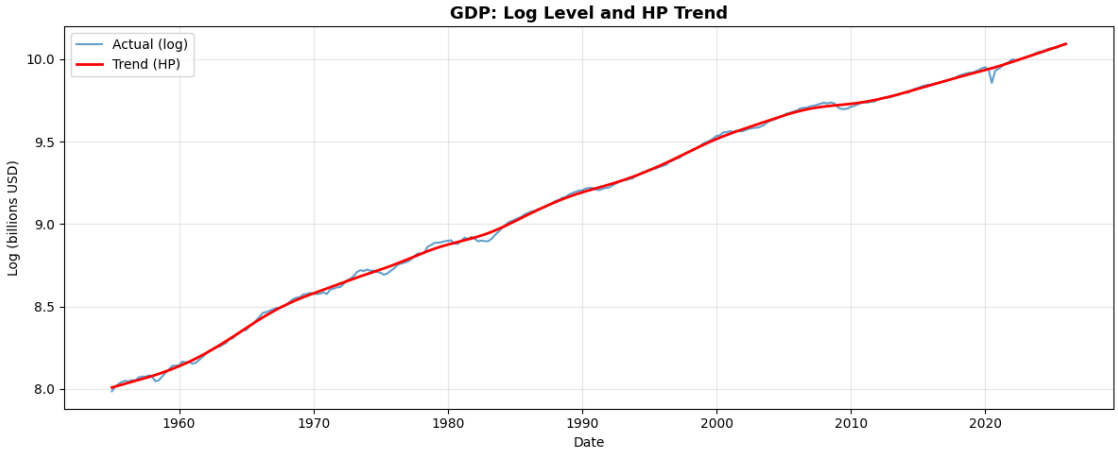
[11]:
```
# Apply HP filter to log series
trend_gdp_log, cycle_gdp_log = hp_manual(y_gdp, lam)
trend_inv_log, cycle_inv_log = hp_manual(y_inv, lam)

print("Descomposición HP completada:")
print(f"  GDP:  trend={trend_gdp_log[0]:.4f}, cycle={cycle_gdp_log[0]:.4f}")
print(f"  Inv:  trend={trend_inv_log[0]:.4f}, cycle={cycle_inv_log[0]:.4f}")
print(f"Std(cycle_gdp): {np.std(cycle_gdp_log):.6f}")
print(f"Std(cycle_inv):  {np.std(cycle_inv_log):.6f}")
print("  Tendencias y ciclos extraídos.")
```

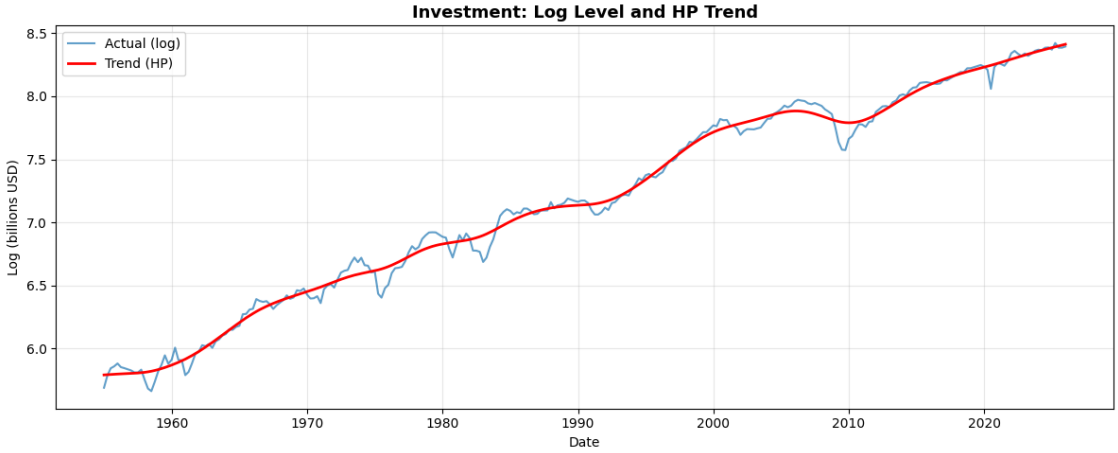
```
Descomposición HP completada:
  GDP:  trend=8.0081, cycle=-0.0230
  Inv:  trend=5.7917, cycle=-0.1017
Std(cycle_gdp): 0.015078
Std(cycle_inv):  0.063306
  Tendencias y ciclos extraídos.
```

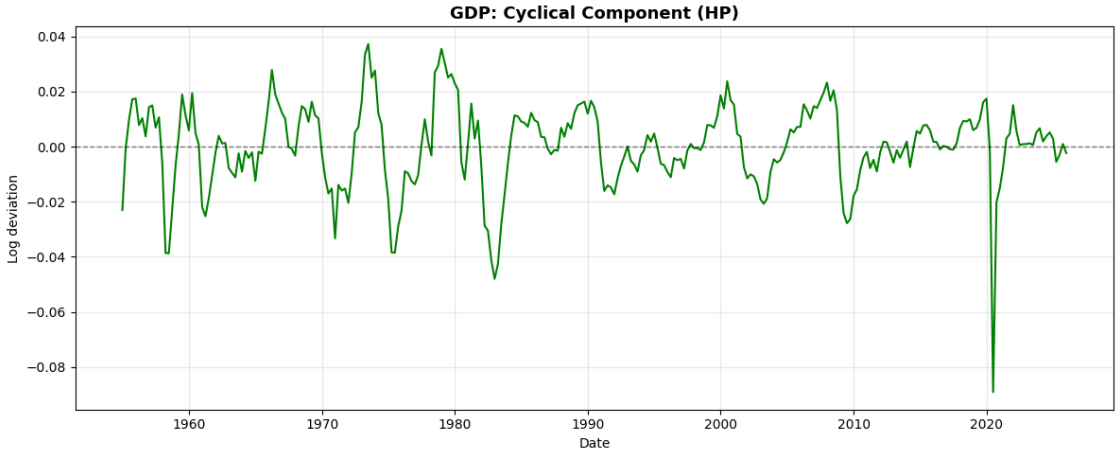
3.4 2.4 Gráficos de Tendencia y Ciclo (Log)

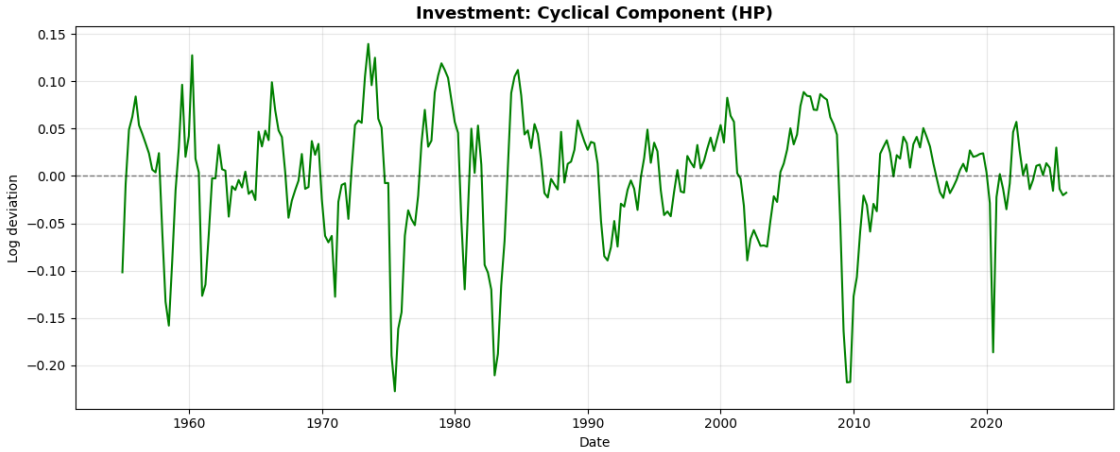
[12]:
```
# Function to create and save trend vs. actual plots
def plot_trend(df, y, trend, title, ylabel, filename):
    fig, ax = plt.subplots(figsize=(12, 5))
    ax.plot(df['date'], y, linewidth=1.5, label='Actual (log)', alpha=0.7)
    ax.plot(df['date'], trend, linewidth=2, label='Trend (HP)', color='red')
    ax.set_xlabel('Date')
    ax.set_ylabel(ylabel)
    ax.set_title(title, fontsize=13, fontweight='bold')
    ax.legend(loc='best')
    ax.grid(True, alpha=0.3)
    plt.tight_layout()
    plt.savefig(OUT_FIG / filename, dpi=150, bbox_inches='tight')
    plt.show()
    plt.close()
```

```python
# Function to create and save cycle plots
def plot_cycle(df, cycle, title, ylabel, filename):
    fig, ax = plt.subplots(figsize=(12, 5))
    ax.plot(df['date'], cycle, linewidth=1.5, color='green')
    ax.axhline(y=0, color='black', linestyle='--', linewidth=1, alpha=0.5)
    ax.set_xlabel('Date')
    ax.set_ylabel(ylabel)
    ax.set_title(title, fontsize=13, fontweight='bold')
    ax.grid(True, alpha=0.3)
    plt.tight_layout()
    plt.savefig(OUT_FIG / filename, dpi=150, bbox_inches='tight')
    plt.show()
    plt.close()

# Plot GDP trend vs actual
plot_trend(df, y_gdp, trend_gdp_log,
           'GDP: Log Level and HP Trend', 'Log (billions USD)',
           '08_hp_gdp_trend.png')

# Plot Investment trend vs actual
plot_trend(df, y_inv, trend_inv_log,
           'Investment: Log Level and HP Trend', 'Log (billions USD)',
           '09_hp_inv_trend.png')

# Plot GDP cycle
plot_cycle(df, cycle_gdp_log,
           'GDP: Cyclical Component (HP)', 'Log deviation',
           '10_hp_gdp_cycle.png')

# Plot Investment cycle
plot_cycle(df, cycle_inv_log,
           'Investment: Cyclical Component (HP)', 'Log deviation',
           '11_hp_inv_cycle.png')

print("  4 gráficos de tendencia y ciclo guardados.")
```


GDP: Log Level and HP Trend
Actual (log)
Trend (HP)
Log (billions USD)
Date


Investment: Log Level and HP Trend
Actual (log)
Trend (HP)
Log (billions USD)
Date


GDP: Cyclical Component (HP)
Log deviation
Date




```
4 gráficos de tendencia y ciclo guardados.
```

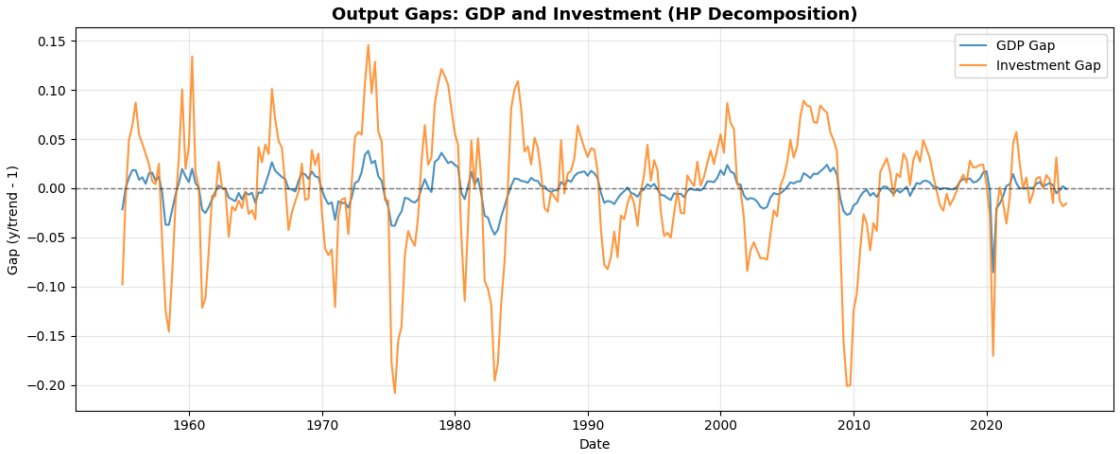
3.5 2.5 Brechas en Niveles: Output Gap (y/trend - 1)

[13]:
```
# For gap computation (y/trend - 1), we need HP in levels, not log
gdp_level = df["GDP"].to_numpy()
inv_level = df["investment"].to_numpy()

# Apply HP filter in levels
trend_gdp_lvl, _ = hp_manual(gdp_level, lam)
trend_inv_lvl, _ = hp_manual(inv_level, lam)

# Compute output gaps
gap_gdp = gap_ratio(gdp_level, trend_gdp_lvl)
gap_inv = gap_ratio(inv_level, trend_inv_lvl)

print("Brechas calculadas (niveles):")
print(f"  gap_gdp:  min={gap_gdp.min():.6f}, mean={gap_gdp.mean():.6f},␣
  ↪max={gap_gdp.max():.6f}")
print(f"  gap_inv:  min={gap_inv.min():.6f}, mean={gap_inv.mean():.6f},␣
  ↪max={gap_inv.max():.6f}")
print("  Brechas del PIB e Inversión calculadas.")
```

```
Brechas calculadas (niveles):
  gap_gdp:  min=-0.085556, mean=-0.000063, max=0.038023
  gap_inv:  min=-0.208433, mean=-0.000875, max=0.145733
  Brechas del PIB e Inversión calculadas.
```

3.6 2.6 Gráfico Conjunto: Brechas de GDP e Investment

[14]:
```
# Plot combined gaps
fig, ax = plt.subplots(figsize=(12, 5))
ax.plot(df['date'], gap_gdp, linewidth=1.5, label='GDP Gap', alpha=0.8)
ax.plot(df['date'], gap_inv, linewidth=1.5, label='Investment Gap', alpha=0.8)
ax.axhline(y=0, color='black', linestyle='--', linewidth=1, alpha=0.5)
ax.set_xlabel('Date')
ax.set_ylabel('Gap (y/trend - 1)')
ax.set_title('Output Gaps: GDP and Investment (HP Decomposition)', fontsize=13,␣
 ↪fontweight='bold')
ax.legend(loc='best')
ax.grid(True, alpha=0.3)
plt.tight_layout()
plt.savefig(OUT_FIG / '12_hp_gaps_gdp_inv.png', dpi=150, bbox_inches='tight')
plt.show()
plt.close()

print("  Gráfico de brechas combinadas guardado.")
```




```
  Gráfico de brechas combinadas guardado.
```

3.7 2.7 Interpretación: Tendencia, Ciclo y Brechas (HP manual)

3.7.1 2.7.1 Qué estamos midiendo (log vs niveles)

En esta fase calculamos dos objetos relacionados pero no idénticos:

1) **Descomposición en log-niveles**: aplicamos HP a $\log(y_t)$ para obtener una tendencia suave y un ciclo interpretado como **desviación porcentual aproximada** (porque para variaciones pequeñas, $\log(y_t) - \log(\tau_t) \approx (y_t - \tau_t)/\tau_t$).

2) **Brecha en niveles (definición del enunciado)**: construimos la brecha como

$$\text{gap}_t = \frac{y_t}{\tau_t} - 1,$$

donde y_t es el nivel observado (GDP o inversión) y τ_t es su tendencia HP estimada en niveles.

Ambas lecturas deben contar una historia coherente: el ciclo en log y la brecha en niveles capturan el mismo fenómeno (fluctuaciones alrededor de tendencia), pero en escalas ligeramente distintas.

3.7.2 2.7.2 Tendencia de largo plazo: PIB vs inversión

En los gráficos de log(GDP) y log(investment) con su tendencia HP, la tendencia del PIB es muy suave y creciente, lo cual refleja el componente secular de crecimiento. En inversión, la tendencia también crece, pero el ajuste HP sugiere cambios más marcados en la pendiente en algunos tramos, consistente con que la inversión sufre reajustes más pronunciados a lo largo del tiempo.

3.7.3 2.7.3 Ciclo: amplitud y sensibilidad (hecho estilizado clave)

Los ciclos (componente cíclico de HP en log) muestran una regularidad central: - **La inversión presenta oscilaciones de mayor amplitud** que el PIB. - En términos de brecha en niveles, esto también se confirma: el rango del gap del PIB es aproximadamente [**-0.0856, 0.0380**], mientras que el de inversión es [**-0.2084, 0.1457**].

Interpretación económica: la inversión es el componente del gasto agregado más sensible al ciclo porque responde con mayor elasticidad a expectativas, incertidumbre y condiciones financieras; por eso, ante shocks negativos, suele caer más que el PIB agregado y también recuperarse con mayor fuerza.

3.7.4 2.7.4 Episodios cíclicos visibles (anclados a evidencia)

El comportamiento alrededor de 2020 es el episodio más claro en ambas medidas. En **2020-Q2 (2020-06-30)**, las brechas son marcadamente negativas: - **GDP gap = -0.0856**, es decir, el PIB se ubica aproximadamente **8.6% por debajo** de su tendencia HP en ese trimestre. - **Investment gap = -0.1705**, es decir, la inversión se ubica aproximadamente **17.1% por debajo** de su tendencia.

En **2020-Q3 (2020-09-30)** se observa una reversión rápida hacia valores cercanos a cero: - **GDP gap = -0.0204** (2.0% por debajo de tendencia), - **Investment gap = -0.0230** (2.3% por debajo de tendencia).

Esto resume un patrón consistente con shocks macroeconómicos abruptos: una caída muy fuerte seguida por una recuperación parcial rápida. Además, la gráfica conjunta de brechas muestra co-movimiento: en varias fases del tiempo las brechas tienden a moverse en la misma dirección, con inversión usualmente amplificando la magnitud del ciclo del PIB.

De forma complementaria, los extremos de la muestra refuerzan esta lectura: el mínimo del PIB ocurre en **2020-06-30** con **-0.0856**, mientras que la inversión alcanza un mínimo más profundo en **1975-06-30** con **-0.2084**. En expansiones, ambos máximos coinciden en **1973-06-30**: **GDP gap = 0.0380** e **investment gap = 0.1457**, lo que sugiere que la inversión también reacciona con mayor magnitud al alza en fases expansivas.

3.7.5 2.7.5 Cómo leer el signo (y por qué importa)

- $\text{gap}_t > 0$: la variable está por encima de su tendencia → fase expansiva.
- $\text{gap}_t < 0$: por debajo de tendencia → fase contractiva / desaceleración.
- $\text{gap}_t \approx 0$: alineada con la tendencia.

En la práctica, el tamaño importa: brechas pequeñas (cercanas a cero) indican fluctuaciones moderadas; brechas grandes en valor absoluto señalan episodios excepcionalmente pronunciados.

3.7.6 2.7.6 Nota técnica mínima: sesgo en los extremos (end-point bias)

El filtro HP es conocido por sufrir **sesgo en los extremos de la muestra**: la tendencia estimada cerca del inicio y del final puede ser menos confiable porque el filtro no dispone de datos futuros ni pasados. Por tanto, las brechas muy cerca del final de la muestra deben interpretarse con cautela, evitando sobre-interpretación de fluctuaciones puntuales. Sin embargo, los argumentos centrales descansan en episodios robustos claramente visibles, como el choque de 2020-Q2, cuya magnitud y reversión rápida evidencian un fenómeno cíclico genuino.

4 3 Fase 3 — HP con librería y hechos estilizados (statsmodels)

En esta fase utilizamos una implementación estándar del filtro Hodrick-Prescott a través de la librería `statsmodels` para garantizar comparabilidad con análisis macroeconómicos convencionales. Mantenemos la definición de brecha en niveles: gap = (y/) - 1, donde y es la serie original y es la tendencia extraída por el filtro HP con = 1600 (convención para datos trimestrales).

4.1 3.1 Brechas HP (statsmodels) para las 6 variables

[15]:
```
import sys
import pandas as pd
from pathlib import Path

# Asegurar import robusto de src.cycle_stats
if str(ROOT) not in sys.path:
    sys.path.insert(0, str(ROOT))

from src.cycle_stats import hp_library_gap

print("  Importaciones completadas: hp_library_gap desde src.cycle_stats")
```

```
 Importaciones completadas: hp_library_gap desde src.cycle_stats
```

[16]:
```
# Parámetro de suavización
lam = 1600

# Construir DataFrame con brechas HP usando statsmodels para las 6 variables
variable_mapping = {
    'GDP': 'GDP',
    'consumption': 'consumption',
    'investment': 'investment',
```

```
    'unemployment': 'unemployment',
    'inflation': 'inflation',
    'fedfunds': 'fedfunds'
}

gaps_df = pd.DataFrame(index=df['date'])

for col_orig, col_name in variable_mapping.items():
    # Calcular brecha con hp_library_gap y renombrar para quitar sufijo _gap
    gap_series = hp_library_gap(df[col_orig], lam=lam).rename(col_name)
    gaps_df[col_name] = gap_series.values

# Convertir index a DatetimeIndex si es necesario
gaps_df.index = pd.to_datetime(gaps_df.index)

print(f" DataFrame gaps_df construido: {gaps_df.shape[0]} filas × {gaps_df.
 ↪shape[1]} columnas")
print(f" Columnas: {list(gaps_df.columns)}")
print(f" Rango de fechas: {gaps_df.index.min().date()} a {gaps_df.index.max().
 ↪date()}")
```

```
 DataFrame gaps_df construido: 285 filas × 6 columnas
 Columnas: ['GDP', 'consumption', 'investment', 'unemployment', 'inflation',
'fedfunds']
 Rango de fechas: 1954-12-31 a 2025-12-31
```

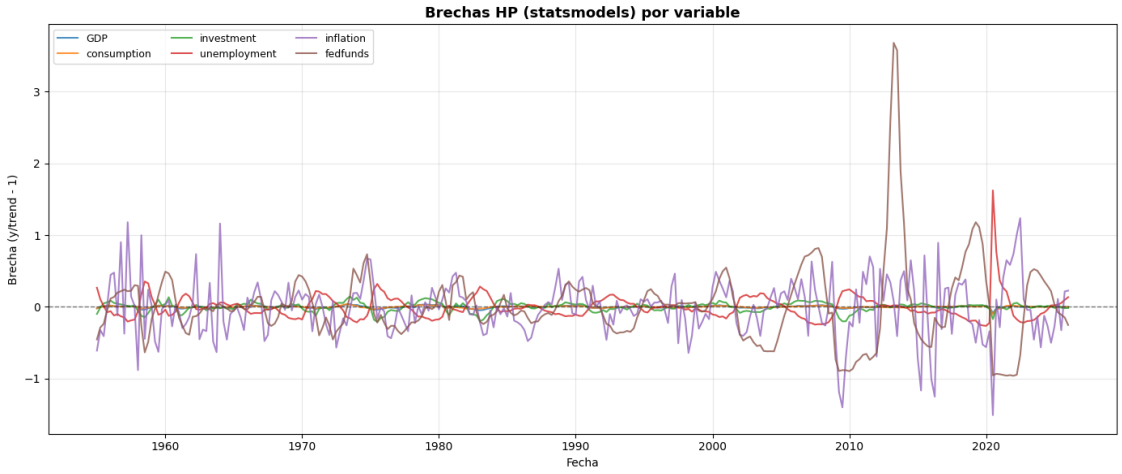
[17]:
```
# Mostrar primeras filas
print("Primeras 5 filas de gaps_df:")
display(gaps_df.head())

# Mostrar estadísticas descriptivas
print("\nEstadísticas descriptivas:")
desc_gaps = gaps_df.describe().round(4)
display(desc_gaps)

# Preparar para exportación con fecha legible (sin hora)
export_df = gaps_df.reset_index()
export_df['date'] = pd.to_datetime(export_df['date']).dt.date

# Guardar a Excel
xlsx_path = OUT_TAB / 'hp_gaps_statsmodels.xlsx'
export_df.to_excel(xlsx_path, index=False, sheet_name='gaps')
print(f"\nGuardado: {xlsx_path.relative_to(ROOT)}")

# Guardar a CSV (opcional)
csv_path = OUT_TAB / 'hp_gaps_statsmodels.csv'
export_df.to_csv(csv_path, index=False)
```

```
print(f"Guardado: {csv_path.relative_to(ROOT)}")
```

```
Primeras 5 filas de gaps_df:
                 GDP  consumption  investment  unemployment  inflation  \
date
1954-12-31 -0.021432    -0.019833   -0.097826      0.266783  -0.608474
1955-03-31  0.000480    -0.005241   -0.009314      0.105562  -0.322860
1955-06-30  0.010710     0.006396    0.049289      0.010722  -0.407755
1955-09-30  0.018328     0.011452    0.063859     -0.073855   0.014692
1955-12-31  0.018476     0.016833    0.087141     -0.059945   0.445722

            fedfunds
date
1954-12-31 -0.453348
1955-03-31 -0.286681
1955-06-30 -0.235081
1955-09-30 -0.047810
1955-12-31  0.116202


Estadísticas descriptivas:
              GDP  consumption  investment  unemployment  inflation  fedfunds
count  285.0000     285.0000    285.0000      285.0000   285.0000  285.0000
mean    -0.0001      -0.0001     -0.0009       -0.0054    -0.0152    0.0105
std      0.0150       0.0132      0.0616        0.1666     0.3935    0.5413
min     -0.0856      -0.1043     -0.2084       -0.2649    -1.5072   -0.9595
25%     -0.0076      -0.0071     -0.0278       -0.1024    -0.2477   -0.2615
50%      0.0005       0.0008      0.0043       -0.0268    -0.0070   -0.0314
75%      0.0094       0.0080      0.0385        0.0598     0.2016    0.2251
max      0.0380       0.0376      0.1457        1.6213     1.2356    3.6762


Guardado: outputs/tables/hp_gaps_statsmodels.xlsx
Guardado: outputs/tables/hp_gaps_statsmodels.csv
```

[18]:
```
# Figura 13: Brechas HP (statsmodels) por variable

fig, ax = plt.subplots(figsize=(14, 6))

for col in gaps_df.columns:
    ax.plot(gaps_df.index, gaps_df[col], linewidth=1.5, label=col, alpha=0.8)

ax.axhline(y=0, color='black', linestyle='--', linewidth=1, alpha=0.5)
ax.set_xlabel('Fecha')
ax.set_ylabel('Brecha (y/trend - 1)')
ax.set_title('Brechas HP (statsmodels) por variable', fontsize=13,␣
 ↪fontweight='bold')
```

```
ax.legend(ncol=3, fontsize=9)
ax.grid(True, alpha=0.3)

plt.tight_layout()
fig_path = OUT_FIG / "13_hp_gaps_statsmodels.png"
plt.savefig(fig_path, dpi=150, bbox_inches='tight')
plt.show()
plt.close()
```



4.2 3.2 Volatilidad y volatilidad relativa (brechas HP)

La desviación estándar de cada brecha (std) mide la magnitud del ciclo económico alrededor de la tendencia. La volatilidad relativa expresa qué tan volátil es cada variable en comparación con el PIB: valores mayores a 1 indican que la variable amplifica el ciclo del PIB, mientras que valores menores a 1 indican que amortigua las fluctuaciones económicas.

[19]:
```
from src.cycle_stats import volatility_table

# Calcular tabla de volatilidad
vol_df = volatility_table(gaps_df, gdp_col="GDP")

# Mostrar en el notebook
print("Tabla de Volatilidad (Desviación Estándar y Volatilidad Relativa):")
display(vol_df.round(4))

# Guardar a Excel
xlsx_path = OUT_TAB / 'volatilidad_hp.xlsx'
vol_df.to_excel(xlsx_path, sheet_name='volatilidad')
print(f"\nGuardado: {xlsx_path.relative_to(ROOT)}")

# Guardar a CSV (opcional)
```

```
csv_path = OUT_TAB / 'volatilidad_hp.csv'
vol_df.to_csv(csv_path)
print(f"Guardado: {csv_path.relative_to(ROOT)}")
```

```
Tabla de Volatilidad (Desviación Estándar y Volatilidad Relativa):

                std  std_rel_to_gdp
GDP          0.0150          1.0000
consumption  0.0132          0.8793
investment   0.0616          4.1036
unemployment 0.1666         11.0954
inflation    0.3935         26.1985
fedfunds     0.5413         36.0366


Guardado: outputs/tables/volatilidad_hp.xlsx
Guardado: outputs/tables/volatilidad_hp.csv
```

4.3 3.3 Correlaciones contemporáneas con brecha del PIB

La correlación contemporánea mide co-movimiento simultáneo entre la brecha de cada variable y la brecha del PIB. Correlaciones positivas indican comportamiento procíclico (amplificador del ciclo), correlaciones negativas indican contracíclico (amortiguador), y cercanas a cero sugieren movimientos acíclicos (independientes del ciclo del PIB).

[20]:
```
from src.cycle_stats import corr_table

# Calcular tabla de correlaciones contemporáneas
corr_df = corr_table(gaps_df, gdp_col="GDP")

# Mostrar en el notebook
print("Tabla de Correlaciones Contemporáneas (con brecha del PIB):")
display(corr_df.round(4))

# Guardar a Excel
xlsx_path = OUT_TAB / 'correlaciones_hp.xlsx'
corr_df.to_excel(xlsx_path, sheet_name='correlaciones')
print(f"\nGuardado: {xlsx_path.relative_to(ROOT)}")

# Guardar a CSV (opcional)
csv_path = OUT_TAB / 'correlaciones_hp.csv'
corr_df.to_csv(csv_path)
print(f"Guardado: {csv_path.relative_to(ROOT)}")
```

```
Tabla de Correlaciones Contemporáneas (con brecha del PIB):

             corr_with_gdp
consumption         0.8811
investment          0.8893
unemployment       -0.7985
```

```
inflation            0.2224
fedfunds             0.3104


Guardado: outputs/tables/correlaciones_hp.xlsx
Guardado: outputs/tables/correlaciones_hp.csv
```

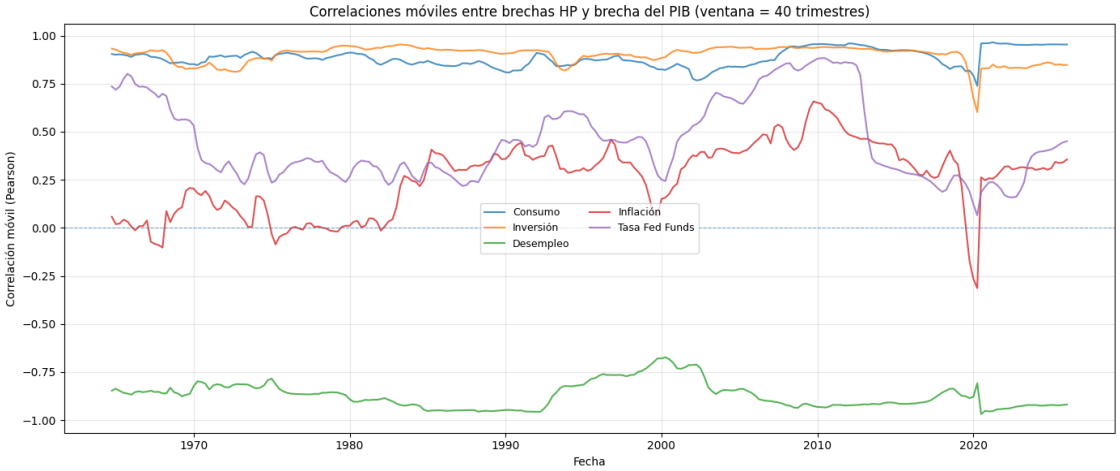
4.4 3.4 Correlaciones móviles (rolling) con ventana de 40 trimestres

Las correlaciones móviles permiten estudiar la estabilidad temporal de las relaciones cíclicas. Una ventana de 40 trimestres (aproximadamente 10 años) captura dinámicas de mediano plazo sin perder resolución temporal para detectar cambios estructurales.

[21]:
```python
from src.cycle_stats import rolling_corr_table
import matplotlib.pyplot as plt
import pandas as pd
from IPython.display import display

# Definir ventana móvil
window = 40

# Calcular correlaciones móviles
roll_df = rolling_corr_table(gaps_df, window=window, gdp_col="GDP")

# Mostrar tabla (en notebook) - compacta y legible
display(roll_df.tail(5).round(4))

# Guardar a Excel
xlsx_path = OUT_TAB / "correlaciones_moviles_hp.xlsx"
roll_df.to_excel(xlsx_path, sheet_name="correlaciones_moviles")
print(f"Guardado: {xlsx_path.relative_to(ROOT)}")

# Guardar a CSV (opcional)
csv_path = OUT_TAB / "correlaciones_moviles_hp.csv"
roll_df.to_csv(csv_path)
print(f"Guardado: {csv_path.relative_to(ROOT)}")

# Mapeo de nombres legibles para la leyenda (en español si tu documento está en␣
 ↪español)
label_mapping = {
    "corr_consumption_gdp": "Consumo",
    "corr_investment_gdp": "Inversión",
    "corr_unemployment_gdp": "Desempleo",
    "corr_inflation_gdp": "Inflación",
    "corr_fedfunds_gdp": "Tasa Fed Funds",
}

# Crear figura con todas las columnas de roll_df
```

```
fig, ax = plt.subplots(figsize=(14, 6))

for col in roll_df.columns:
    label = label_mapping.get(col, col)
    ax.plot(roll_df.index, roll_df[col], label=label, linewidth=1.5, alpha=0.8)

ax.set_xlabel("Fecha")
ax.set_ylabel("Correlación móvil (Pearson)")
ax.set_title(f"Correlaciones móviles entre brechas HP y brecha del PIB (ventana␣
 ↪= {window} trimestres)")
ax.grid(True, alpha=0.3)
ax.axhline(y=0, linestyle="--", linewidth=0.8, alpha=0.6)

ax.legend(ncol=2, fontsize=9, frameon=True)
fig.tight_layout()

# Guardar figura
fig_path = OUT_FIG / "14_rolling_corr.png"
fig.savefig(fig_path, dpi=150)
plt.show()
plt.close(fig)
```

```
            corr_consumption_gdp  corr_investment_gdp  corr_unemployment_gdp  \
date
2024-12-31                0.9544               0.8579                -0.9216
2025-03-31                0.9541               0.8477                -0.9221
2025-06-30                0.9545               0.8504                -0.9234
2025-09-30                0.9540               0.8477                -0.9206
2025-12-31                0.9540               0.8475                -0.9189

            corr_inflation_gdp  corr_fedfunds_gdp
date
2024-12-31              0.3108             0.4080
2025-03-31              0.3434             0.4190
2025-06-30              0.3372             0.4332
2025-09-30              0.3408             0.4446
2025-12-31              0.3565             0.4510

Guardado: outputs/tables/correlaciones_moviles_hp.xlsx
Guardado: outputs/tables/correlaciones_moviles_hp.csv
```



Nota: La correlación móvil se calcula con una ventana de 40 trimestres (10 años), por lo que los primeros 39 puntos son `NaN` por construcción.
Lectura: valores positivos indican co-movimiento procíclico con la brecha del PIB; valores negativos, comportamiento contracíclico.

4.5 3.5 Interpretación (hechos estilizados)

La tabla de volatilidad muestra que la **amplitud cíclica** (desviación estándar de la brecha HP) difiere sustancialmente entre variables. En particular, la **inversión** presenta una volatilidad claramente superior a la del PIB: su volatilidad relativa es **4.10×** (std 0.0616 vs 0.0150 en PIB), consistente con el hecho estilizado de que la inversión **amplifica** el ciclo. En contraste, **consumo** es menos volátil que el PIB (0.88×). En variables en tasa, las brechas aparecen con volatilidad relativa alta (**desempleo 11.10×**, **inflación 26.20×**, **fed funds 36.04×**); esto debe interpretarse con cautela, porque en estas series el HP puede capturar no solo fluctuaciones cíclicas sino también **cambios de régimen** en el nivel/tendencia.

En correlaciones contemporáneas con la brecha del PIB, **consumo** e **inversión** resultan claramente **procíclicos** (corr **0.881** y **0.889**, respectivamente), mientras que **desempleo** es **contracíclico** (corr **−0.799**). Este patrón es coherente con la lectura estándar del mercado laboral: cuando el producto está por debajo de tendencia, el desempleo tiende a ubicarse por encima de su nivel tendencial. En cambio, **inflación** y **fed funds** presentan correlaciones contemporáneas positivas pero moderadas (**0.222** y **0.310**), sugiriendo una relación menos estable/intensa con el ciclo medido por la brecha del PIB.

Finalmente, las correlaciones móviles (ventana de **40 trimestres 10 años**) permiten evaluar la estabilidad temporal de estos co-movimientos. El gráfico sugiere que el co-movimiento procíclico de **consumo** e **inversión** con el PIB es alto y relativamente estable en la mayor parte de la muestra (promedios móviles **0.884** y **0.901**), mientras que el **desempleo** permanece fuertemente contracíclico (promedio móvil **−0.871**). En contraste, **inflación** y **fed funds** exhiben mayor variación: la correlación móvil de inflación incluso cambia de signo en algunos tramos (p. ej., cae por debajo de cero alrededor de 2020), lo cual es consistente con que la relación entre inflación, política monetaria y ciclo depende del régimen macroeconómico. Estas correlaciones describen asociación dinámica y **no** deben interpretarse como causalidad.

5 4 Fase 4 — Matriz insumo-producto (I-O)

La matriz insumo-producto es un registro sistemático de los flujos de transacciones intermedias entre todos los sectores de una economía. Denotamos por Z la matriz de transacciones intermedias (insumos que cada sector compra de otros), y por x el vector de output total por sector. Los coeficientes técnicos A_ij = Z_ij / x_j indican cuánta unidad del insumo i se requiere para producir una unidad de output en el sector j. La matriz de Leontief L = (I - A)^(-1) captura los requerimientos totales (directos e indirectos) ante un cambio de una unidad en la demanda final. El análisis de eslabonamiento distingue entre enlaces "backward" (cuánto un sector demanda de insumos de otros, arrastrando demanda hacia sus proveedores) y "forward" (cuánto otros sectores demandan del sector en cuestión, haciendo de él un proveedor clave).

[22]:
```
import sys
sys.path.insert(0, str(src_path))

from io_tools import (load_io_matrix, technical_coefficients, leontief_inverse,
                      backward_forward_direct, backward_forward_indirect,␣
 ↪rank_linkages)

# Cargar matriz insumo-producto
io_path = DATA_RAW / "Matriz.xlsx"
Z, x, sectors = load_io_matrix(io_path)
n = len(sectors)

# Construir matrices de coeficientes técnicos y Leontief
A = technical_coefficients(Z, x)
L = leontief_inverse(A)
I = np.eye(n)

print(f"Matriz Z cargada: {Z.shape}")
print(f"Vector x: {x.shape}")
print(f"Matriz A (coeficientes técnicos): {A.shape}")
print(f"Matriz L (inversa Leontief): {L.shape}")
print(f"\nSectores: {n}")
print(f"Primeros 5: {sectors[:5]}")
print(f"Últimos 5: {sectors[-5:]}")
```

```
Matriz Z cargada: (20, 20)


Vector x: (20,)
Matriz A (coeficientes técnicos): (20, 20)
Matriz L (inversa Leontief): (20, 20)

Sectores: 20
Primeros 5: ['S1_Agro', 'S2_Minas', 'S3_Alimentos', 'S4_Textiles', 'S5_Madera']
Últimos 5: ['S16_Hoteles', 'S17_Comunicaciones', 'S18_Finanzas',
'S19_Inmobiliario', 'S20_EduSalud']
```

5.1 4.1 Eslabonamientos directos (matriz A)

[23]:
```
# Eslabonamientos directos
back_d, forw_d = backward_forward_direct(A)

# Rankings
rank_back_d = rank_linkages(back_d, sectors, top=10)
rank_forw_d = rank_linkages(forw_d, sectors, top=10)

display(rank_back_d)
print()
display(rank_forw_d)
```

	sector	value
0	S11_Vehiculos	0.209517
1	S13_Construccion	0.208265
2	S1_Agro	0.196207
3	S18_Finanzas	0.193871
4	S14_Comercio	0.140131
5	S19_Inmobiliario	0.130181
6	S8_Plasticos	0.110088
7	S12_ServBasicos	0.107832
8	S3_Alimentos	0.099010
9	S4_Textiles	0.088989

	sector	value
0	S12_ServBasicos	0.224178
1	S18_Finanzas	0.196457
2	S14_Comercio	0.166980
3	S7_Quimicos	0.158379
4	S15_Transporte	0.154070
5	S9_Metales	0.134232
6	S10_Maquinaria	0.122326
7	S17_Comunicaciones	0.117572
8	S19_Inmobiliario	0.104257
9	S13_Construccion	0.101385

5.2 4.2 Eslabonamientos totales (directos + indirectos) usando L

[24]:
```
# Eslabonamientos totales (directos + indirectos)
back_t, forw_t = backward_forward_indirect(L)

# Rankings
rank_back_t = rank_linkages(back_t, sectors, top=10)
rank_forw_t = rank_linkages(forw_t, sectors, top=10)

display(rank_back_t)
```

```
print()
display(rank_forw_t)
```

```
             sector     value
0     S11_Vehiculos  1.237554
1  S13_Construccion  1.236254
2           S1_Agro  1.222458
3      S18_Finanzas  1.221247
4      S14_Comercio  1.158823
5  S19_Inmobiliario  1.150192
6      S8_Plasticos  1.123545
7   S12_ServBasicos  1.121080
8      S3_Alimentos  1.113353
9       S4_Textiles  1.100247
```

```
               sector     value
0     S12_ServBasicos  1.253836
1        S18_Finanzas  1.225193
2        S14_Comercio  1.188533
3         S7_Quimicos  1.177877
4      S15_Transporte  1.173855
5          S9_Metales  1.151267
6      S10_Maquinaria  1.138516
7  S17_Comunicaciones  1.134585
8    S19_Inmobiliario  1.118271
9    S13_Construccion  1.114779
```

5.3 4.3 Eslabonamientos indirectos puros (L - I)

La matriz L incluye unos en la diagonal principal, representando el efecto propio o directo que cada sector tiene en sí mismo (una unidad adicional de demanda final genera exactamente una unidad de output sectorial). Al restar la identidad I de L, obtenemos L - I, que aísla únicamente los efectos indirectos en cadena, es decir, los arrastres y propagaciones que un sector experimenta o genera a través de sus encadenamientos con otros sectores.

[25]:
```
# Eslabonamientos indirectos puros (L - I)
L_ind = L - I
back_ind = L_ind.sum(axis=0)
forw_ind = L_ind.sum(axis=1)

# Rankings
rank_back_ind = rank_linkages(back_ind, sectors, top=10)
rank_forw_ind = rank_linkages(forw_ind, sectors, top=10)

display(rank_back_ind)
print()
display(rank_forw_ind)
```

```
             sector     value
0     S11_Vehiculos  0.237554
1  S13_Construccion  0.236254
2           S1_Agro  0.222458
3      S18_Finanzas  0.221247
4      S14_Comercio  0.158823
5  S19_Inmobiliario  0.150192
6      S8_Plasticos  0.123545
7   S12_ServBasicos  0.121080
8      S3_Alimentos  0.113353
9       S4_Textiles  0.100247
```

```
               sector     value
0     S12_ServBasicos  0.253836
1        S18_Finanzas  0.225193
2        S14_Comercio  0.188533
3         S7_Quimicos  0.177877
4      S15_Transporte  0.173855
5          S9_Metales  0.151267
6      S10_Maquinaria  0.138516
7  S17_Comunicaciones  0.134585
8    S19_Inmobiliario  0.118271
9    S13_Construccion  0.114779
```

[26]:
```python
# Guardar resultados a Excel
output_file = OUT_TAB / "io_linkages.xlsx"

with pd.ExcelWriter(output_file, engine='openpyxl') as writer:
    rank_back_d.to_excel(writer, sheet_name='A_backward_direct_top10',␣
 ↪index=False)
    rank_forw_d.to_excel(writer, sheet_name='A_forward_direct_top10',␣
 ↪index=False)
    rank_back_t.to_excel(writer, sheet_name='L_backward_total_top10',␣
 ↪index=False)
    rank_forw_t.to_excel(writer, sheet_name='L_forward_total_top10',␣
 ↪index=False)
    rank_back_ind.to_excel(writer, sheet_name='L-I_backward_indirect_top10',␣
 ↪index=False)
    rank_forw_ind.to_excel(writer, sheet_name='L-I_forward_indirect_top10',␣
 ↪index=False)

print(f"Guardado: {output_file.relative_to(ROOT)}")
```

```
Guardado: outputs/tables/io_linkages.xlsx
```

5.4 4.4 Interpretación económica (matriz insumo-producto)

5.4.1 4.4.1 Lectura económica de los eslabonamientos (qué mide cada ranking)

En una matriz insumo-producto, los **eslabonamientos backward** capturan el potencial de un sector para **arrastrar demanda hacia atrás** (hacia sus proveedores) cuando aumenta su producción/demanda final: un backward alto significa que ese sector requiere relativamente muchos insumos intermedios del resto de la economía. Por su parte, los **eslabonamientos forward** capturan el potencial de un sector para **transmitir shocks hacia adelante** (hacia sus usuarios): un forward alto indica que muchos sectores dependen de él como proveedor de insumos.

En este ejercicio se reportan tres conceptos distintos:

- **Directo (matriz (A))**: encadenamientos de "primera ronda". Mide dependencias inmediatas de insumos (sin multiplicadores).
- **Total (matriz (L))**: encadenamientos **directos + indirectos** (con multiplicadores), incorporando todas las rondas de retroalimentación en la cadena productiva.
- **Indirecto puro ((L - I))**: encadenamientos netos excluyendo el efecto propio (la "autodemanda" implícita), para aislar mejor la interdependencia con el resto del sistema.

5.4.2 4.4.2 Resultados en "bloques" sectoriales (lectura estructural)

Con base en los Top 5 reportados, se observan tres bloques interpretables:

(i) Bloque "manufactura durable + construcción" como motores de arrastre (backward alto)
En los eslabonamientos **backward directos ((A))** aparecen **S11_Vehículos**, **S13_Construcción** y **S1_Agro** como sectores con fuerte capacidad de arrastre. Esto sugiere que, cuando estos sectores expanden actividad, demandan una canasta amplia de insumos intermedios (materiales, logística, servicios empresariales, etc.), generando un efecto multiplicador "hacia atrás" sobre proveedores.

(ii) Bloque "servicios básicos + logística + químicos" como infraestructura de provisión (forward alto)
En los eslabonamientos **forward directos ((A))** lidera **S12_Servicios Básicos**, seguido por **S14_Comercio**, **S15_Transporte** y **S7_Químicos**. La lectura es clara: son sectores que funcionan como **insumos transversales** para múltiples actividades (servicios esenciales, distribución comercial, transporte y logística, e insumos químicos para procesos productivos). Por lo tanto, shocks en estos sectores tienden a "irradiarse" hacia muchas ramas usuarias.

(iii) Intermediación y circulación: el rol recurrente de Finanzas y Comercio
S18_Finanzas y **S14_Comercio** aparecen repetidamente en rankings directos y totales. Esto es consistente con una economía donde la **intermediación** (crédito, servicios financieros) y la **circulación** (comercio/distribución) operan como nodos centrales: conectan muchos sectores, no solo por su tamaño sino por su posición en la red de transacciones.

5.4.3 4.4.3 Directo vs total vs indirecto puro: qué cambia al pasar de (A) a (L)

Al pasar de **(A)** (directo) a **(L)** (total), los mismos sectores "centrales" tienden a mantenerse en posiciones altas, pero ahora el ranking refleja **multiplicadores**: no solo "quién compra a quién", sino también las rondas sucesivas (proveedor del proveedor, etc.). Por ejemplo, si Vehículos de-

manda insumos, esos insumos demandan energía, transporte y servicios, y así sucesivamente: eso es lo que incorpora (L).

Reportar adicionalmente el **indirecto puro ((L - I))** es útil porque elimina el componente trivial del "efecto propio" y permite comparar sectores por su interdependencia neta con el resto de la economía. En tus resultados, el hecho de que los Top 5 del total y del indirecto puro sean muy similares sugiere que el liderazgo de esos sectores no se debe solo a auto-efectos, sino a encadenamientos reales con el sistema productivo.

5.4.4 4.4.4 Implicaciones macro: 3 shocks hipotéticos y su propagación

A continuación propongo tres shocks (demanda/oferta) y su transmisión según los eslabonamientos:

1. **Shock de demanda: boom de inversión en construcción (S13_Construcción)**
   Si aumenta la demanda final por construcción (por ejemplo, plan de infraestructura), el backward alto implica un arrastre fuerte sobre sectores proveedores: materiales, manufacturas, transporte/comercio y servicios asociados. En términos macro, esto tiende a amplificar la expansión vía multiplicadores de demanda intermedia.

2. **Shock de demanda: expansión del consumo durable asociado a vehículos (S11_Vehículos)**
   Un aumento de demanda por vehículos detona una cadena de requerimientos intermedios relativamente amplia. Con S11_Vehículos en la cima del backward (directo y total), el modelo sugiere un efecto multiplicador significativo sobre proveedores, lo que se traduciría en una expansión más intensa en sectores upstream.

3. **Shock de oferta: disrupción en servicios básicos o logística (S12_ServBasicos / S15_Transporte)**
   Un shock adverso de oferta (p. ej., restricción energética o cuello logístico) en sectores con forward alto se transmite "hacia adelante" porque muchos sectores dependen de esos insumos. El impacto macro suele ser transversal: encarece o limita producción en múltiples ramas, pudiendo generar contracción generalizada incluso si el shock se origina en un solo sector.

5.4.5 4.4.5 Nota final de interpretación (cautelas)

Estos rankings son **medidas de posición en la red productiva**, no de causalidad. Su lectura es comparativa: ayudan a identificar sectores con mayor potencial de propagación de shocks (vía demanda intermedia o provisión de insumos). Los resultados dependen del año base, la cobertura sectorial y la calidad/definición de la matriz IO; por eso conviene interpretarlos como una fotografía estructural coherente con el modelo, más que como una relación invariante en el tiempo.

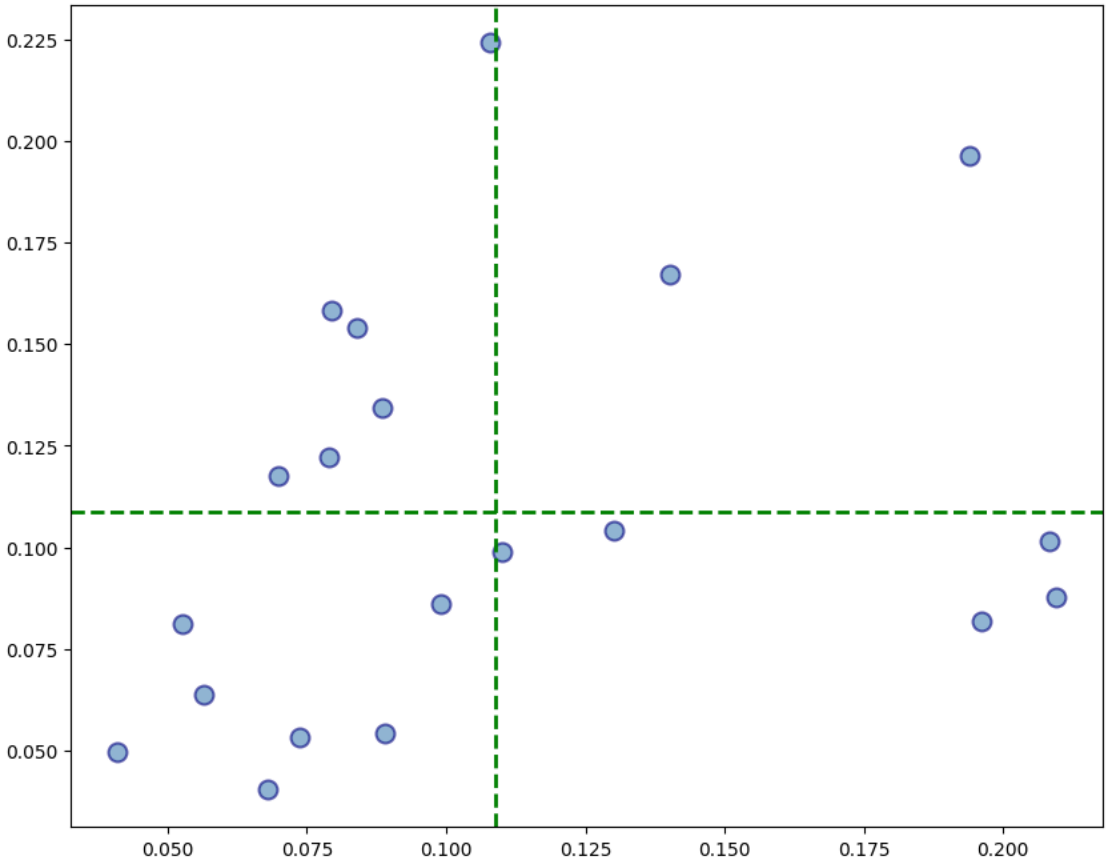
5.5 4.5 Mapa de encadenamientos (Backward vs Forward)

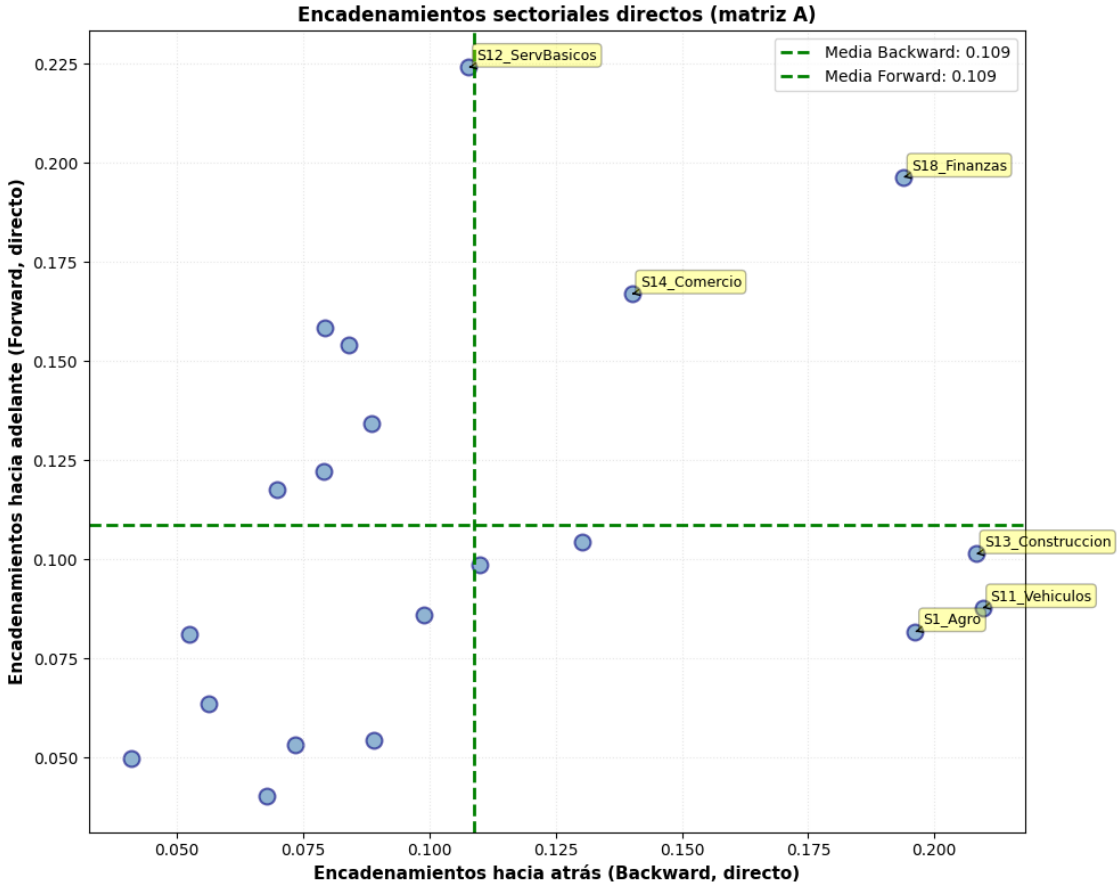
En esta subsección se construye un mapa de dispersión que relaciona los encadenamientos directos hacia atrás (backward) y hacia adelante (forward) de cada sector, basado en la matriz de coeficientes técnicos A. Cada punto representa un sector, y las líneas punteadas indican los promedios de ambas dimensiones. La clasificación resultante en cuadrantes permite identificar sectores clave (alto/alto), impulsores (backward alto), y seguidores (forward alto), facilitando la interpretación de la estructura productiva.

[27]:
```
import matplotlib.pyplot as plt
import numpy as np

# Preparar datos para el gráfico
x_vals = back_d  # Encadenamientos hacia atrás (Backward, directos)
y_vals = forw_d  # Encadenamientos hacia adelante (Forward, directos)

# Crear figura
fig, ax = plt.subplots(figsize=(10, 8))

# Scatter plot
ax.scatter(x_vals, y_vals, s=100, alpha=0.6, color='steelblue',␣
 ↪edgecolors='navy', linewidth=1.5)

# Líneas de corte (promedios)
mean_x = x_vals.mean()
mean_y = y_vals.mean()
ax.axvline(mean_x, color='green', linestyle='--', linewidth=2, label=f'Media␣
 ↪Backward: {mean_x:.3f}')
ax.axhline(mean_y, color='green', linestyle='--', linewidth=2, label=f'Media␣
 ↪Forward: {mean_y:.3f}')

import matplotlib.pyplot as plt
import numpy as np

# Preparar datos para el gráfico
x_vals = back_d  # Encadenamientos hacia atrás (Backward, directos)
y_vals = forw_d  # Encadenamientos hacia adelante (Forward, directos)

# Crear figura
fig, ax = plt.subplots(figsize=(10, 8))

# Scatter plot
ax.scatter(x_vals, y_vals, s=100, alpha=0.6, color='steelblue',␣
 ↪edgecolors='navy', linewidth=1.5)

# Líneas de corte (promedios)
mean_x = x_vals.mean()
mean_y = y_vals.mean()
ax.axvline(mean_x, color='green', linestyle='--', linewidth=2, label=f'Media␣
 ↪Backward: {mean_x:.3f}')
ax.axhline(mean_y, color='green', linestyle='--', linewidth=2, label=f'Media␣
 ↪Forward: {mean_y:.3f}')

# Etiquetas (TOP 3 backward + TOP 3 forward, sin duplicados)
top_back_idx = np.argsort(-x_vals)[:3]
top_forw_idx = np.argsort(-y_vals)[:3]
```

```
unique_idx = sorted(set(list(top_back_idx) + list(top_forw_idx)))

for idx in unique_idx:
    ax.annotate(sectors[idx],
                xy=(x_vals[idx], y_vals[idx]),
                xytext=(5, 5),
                textcoords='offset points',
                fontsize=9,
                bbox=dict(boxstyle='round,pad=0.3', facecolor='yellow', alpha=0.
 ↪3),
                arrowprops=dict(arrowstyle='->', connectionstyle='arc3,rad=0'))

# Etiquetas de ejes y título
ax.set_xlabel('Encadenamientos hacia atrás (Backward, directo)', fontsize=11,␣
 ↪fontweight='bold')
ax.set_ylabel('Encadenamientos hacia adelante (Forward, directo)', fontsize=11,␣
 ↪fontweight='bold')
ax.set_title('Encadenamientos sectoriales directos (matriz A)', fontsize=12,␣
 ↪fontweight='bold')
ax.grid(alpha=0.3, linestyle=':')
ax.legend(loc='best', fontsize=10)

plt.tight_layout()
fig_path = OUT_FIG / '15_io_scatter_direct.png'
fig.savefig(fig_path, dpi=150, bbox_inches='tight')
print(f'Figura guardada: {fig_path.relative_to(ROOT)}')
plt.show()
```

```
Figura guardada: outputs/figures/15_io_scatter_direct.png
```


0.225
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.050
0.075
0.100
0.125
0.150
0.175
0.200



5.6 4.5.2 Interpretación del mapa de encadenamientos (Backward vs Forward, matriz A)

Este gráfico ubica a cada sector en un plano donde:

- Eje **X (Backward directo)**: cuánto "jala" insumos del resto de sectores por cada unidad adicional de producción (demanda de insumos intermedios).
- Eje **Y (Forward directo)**: cuánto "empuja" insumos hacia otros sectores, es decir, qué tan insumo-transversal es como proveedor directo.

Las líneas punteadas corresponden a los **promedios** de backward y forward y dividen el plano en cuatro cuadrantes. Un punto a la derecha/arriba del promedio indica que el sector presenta encadenamientos directos superiores a los del "sector promedio".

5.6.1 4.5.2.1 Por qué las medias de backward y forward coinciden (y por qué no es un error)

En el gráfico, la media backward y la media forward coinciden (o son prácticamente iguales). Esto es esperable bajo la definición estándar:

- Backward directo por sector (j): (b_j= i A{ij}) (suma por columna)

- Forward directo por sector (i): (f_i= j A{ij}) (suma por fila)

Al promediar:

$$\overline{b} = \frac{1}{n} \sum_{j} \sum_{i} A_{ij} \qquad \text{y} \qquad \overline{f} = \frac{1}{n} \sum_{i} \sum_{j} A_{ij}.$$

Ambas cantidades coinciden porque ambas suman **toda la matriz (A)** (solo cambia el orden de la suma). Económicamente, estos promedios resumen la **intensidad promedio de interdependencia directa** del sistema productivo, vista desde el lado "comprador" (backward) y desde el lado "vendedor" (forward). Esta propiedad es útil porque los umbrales vertical y horizontal constituyen una **referencia común** para clasificar sectores por encima o por debajo de la interdependencia directa promedio.

> Nota: con encadenamientos **ponderados** (por tamaño del sector, output o demanda final), esta igualdad podría no cumplirse.

5.6.2 4.5.2.2 Lectura por cuadrantes (interpretación estructural)

Cuadrante I (alto backward, alto forward): "sectores clave / hubs directos"
Estos sectores son simultáneamente demandantes intensivos de insumos y proveedores transversales. En términos de propagación, pueden amplificar shocks tanto **hacia atrás** (a proveedores) como **hacia adelante** (a usuarios). En los resultados, **S18_Finanzas** resalta por ubicarse alto en forward y también alto en backward, consistente con su rol de nodo transversal.

Cuadrante II (bajo backward, alto forward): "infraestructura de insumos / proveedores transversales"
Estos sectores tienden a no demandar tantos insumos intermedios relativos, pero su producción es utilizada por múltiples ramas. Funcionan como "cuellos de botella" potenciales: un shock en ellos se transmite **hacia adelante** con fuerza. En los resultados, **S12_ServBasicos** destaca por su forward elevado, típico de un proveedor de servicios esenciales/insumos generalizados.

Cuadrante III (bajo backward, bajo forward): "sectores periféricos en encadenamientos directos"
Se ubican por debajo del promedio en ambas dimensiones directas. Esto no implica irrelevancia macroeconómica (pueden ser grandes en valor agregado o empleo), sino una menor centralidad como articuladores directos de insumos en la matriz (A).

Cuadrante IV (alto backward, bajo forward): "demandantes intensivos de insumos / sectores de arrastre"
Estos sectores absorben muchos insumos intermedios del resto para producir, pero no necesariamente son proveedores transversales. Su expansión tiende a generar arrastre **hacia atrás** sobre proveedores. En los resultados, **S11_Vehiculos** y **S13_Construccion** aparecen con backward elevado, coherente con cadenas de insumos amplias.

5.6.3 4.5.2.3 Sectores etiquetados: criterio y lectura económica (Top 3 backward / Top 3 forward)

El criterio de etiquetar **los 3 valores más altos** en backward y **los 3 valores más altos** en forward balancea claridad visual y relevancia económica: evita saturación y, al mismo tiempo, identifica los sectores con mayor potencial de propagación directa.

Top backward (arrastre hacia atrás):
- **S11_Vehiculos** y **S13_Construccion**: intensivos en insumos intermedios, por lo que un shock de demanda en estos sectores tiende a elevar la demanda de proveedores.
- **S1_Agro**: backward alto sugiere un uso relativamente intensivo de insumos intermedios directos en su estructura productiva (por ejemplo, insumos químicos, transporte y servicios).

Top forward (irradiación hacia adelante):
- **S12_ServBasicos**: proveedor transversal; shocks en este sector afectan directamente múltiples ramas usuarias.
- **S18_Finanzas**: nodo transversal; cambios en intermediación/servicios financieros tienden a transmitirse de forma amplia.
- **S14_Comercio**: rol de circulación/distribución como insumo operativo para muchas actividades.

5.6.4 4.5.2.4 Implicación macroeconómica: shocks hipotéticos y propagación

1) **Shock de demanda (expansión) en Construcción (S13) o Vehículos (S11)**
   Con backward alto, un aumento de demanda final impulsa una expansión de primera ronda sobre proveedores (materiales, comercio, transporte y servicios asociados). Esto tiende a amplificar el ciclo en sectores upstream.

2) **Shock de oferta adverso en Servicios básicos (S12) o disrupción operativa en Comercio/Transporte (S14/S15)**
   Con forward alto, el shock se transmite a muchas ramas usuarias: un cuello de botella energético/logístico encarece y restringe producción en múltiples sectores, pudiendo generar una contracción agregada amplia incluso si el shock se origina en un solo sector.

3) **Shock financiero (endurecimiento o relajación) asociado a S18_Finanzas**
   La posición alta en forward y relativamente alta en backward sugiere un rol de hub: cambios en intermediación pueden transmitirse de forma amplia tanto a usuarios (condiciones financieras/servicios) como a demandas intermedias asociadas.

5.6.5 4.5.2.5 Cautela final

Este mapa representa **encadenamientos directos** (primera ronda). Para capturar multiplicadores completos y retroalimentaciones se complementa con los resultados basados en (L) (total) y en (L-I) (indirecto puro). Aun así, el gráfico es informativo para identificar sectores con potencial de propagación rápida de shocks en el corto plazo.

6 5 Fase 5 — Empaquetado final (Excel de entrega)

En esta fase final, consolidamos todos los resultados del taller en un único archivo Excel (`outputs/exports/entrega_taller_1.xlsx`) que sirve como entregable completo y autocontenido. Cada hoja corresponde a una etapa de análisis: datos originales, estadísticas descriptivas, descomposiciones HP (manuales y con librería), hechos estilizados e índices de eslabonamiento insumo-producto.

[28]:
```
# Construir Excel final de entrega: entrega_taller_1.xlsx

from openpyxl import load_workbook
```

```
from openpyxl.utils.dataframe import dataframe_to_rows

export_path = OUT_EXP / "entrega_taller_1.xlsx"

with pd.ExcelWriter(export_path, engine='openpyxl') as writer:

    # Hoja 1: data_final
    # Cargar dataset original y exportar con date sin hora
    df_data = pd.read_excel(DATA_PROC / 'dataset_taller.xlsx')
    df_data['date'] = pd.to_datetime(df_data['date']).dt.date
    df_data.to_excel(writer, sheet_name='data_final', index=False)

    # Hoja 2: descriptivas
    desc.to_excel(writer, sheet_name='descriptivas')

    # Hoja 3: gdp_var_negativa
    df_neg['date'] = pd.to_datetime(df_neg['date']).dt.date
    df_neg.to_excel(writer, sheet_name='gdp_var_negativa', index=False)

    # Hoja 4: hp_manual
    # Consolidar series manualmente descompuestas (GDP e Investment)
    hp_manual_df = pd.DataFrame({
        'date': df['date'].dt.date,
        'GDP': df['GDP'],
        'GDP_trend_manual': np.exp(trend_gdp_log),
        'GDP_cycle_manual': cycle_gdp_log,
        'GDP_gap_manual': gap_gdp,
        'investment': df['investment'],
        'investment_trend_manual': np.exp(trend_inv_log),
        'investment_cycle_manual': cycle_inv_log,
        'investment_gap_manual': gap_inv
    })
    hp_manual_df.to_excel(writer, sheet_name='hp_manual', index=False)

    # Hoja 5: hp_lib_gaps
    gaps_export = gaps_df.reset_index()
    gaps_export['date'] = pd.to_datetime(gaps_export['date']).dt.date
    gaps_export.to_excel(writer, sheet_name='hp_lib_gaps', index=False)

    # Hoja 6: hp_lib_stats
    # Escribir volatilidad y correlaciones una debajo de otra
    ws_stats = writer.book.create_sheet('hp_lib_stats')
    row_num = 1

    # Escribir volatilidad
    ws_stats[f'A{row_num}'] = 'VOLATILIDAD (Desviación Estándar y Volatilidad␣
 ↪Relativa)'
```

```
    row_num += 1
    for r_idx, row in enumerate(dataframe_to_rows(vol_df, index=True,␣
↪header=True), row_num):
        for c_idx, value in enumerate(row, 1):
            ws_stats.cell(row=r_idx, column=c_idx, value=value)
    row_num = row_num + len(vol_df) + 2

    # Escribir correlaciones
    ws_stats[f'A{row_num}'] = 'CORRELACIONES CONTEMPORANEAS (con brecha del␣
↪PIB)'
    row_num += 1
    for r_idx, row in enumerate(dataframe_to_rows(corr_df, index=True,␣
↪header=True), row_num):
        for c_idx, value in enumerate(row, 1):
            ws_stats.cell(row=r_idx, column=c_idx, value=value)

    # Hoja 7: rolling_corr
    roll_export = roll_df.reset_index()
    roll_export['date'] = pd.to_datetime(roll_export['date']).dt.date
    roll_export.to_excel(writer, sheet_name='rolling_corr', index=False)

    # Hoja 8-13: io_linkages (6 hojas de I-O)
    io_excel = pd.ExcelFile(OUT_TAB / 'io_linkages.xlsx')
    for sheet_name in io_excel.sheet_names:
        io_df = pd.read_excel(OUT_TAB / 'io_linkages.xlsx',␣
↪sheet_name=sheet_name)
        io_df.to_excel(writer, sheet_name=sheet_name, index=False)

print("Excel final creado: outputs/exports/entrega_taller_1.xlsx")
```

```
Excel final creado: outputs/exports/entrega_taller_1.xlsx
```

[29]:
```
# Verificacion final: confirmar que todos los archivos existen

from pathlib import Path

print("VERIFICACION FINAL DE ENTREGA")
print("=" * 70)

# Archivos Excel principales
excel_files = [
    OUT_EXP / "entrega_taller_1.xlsx",
    OUT_TAB / "io_linkages.xlsx",
]

# Archivos PNG (figuras)
png_files = list(OUT_FIG.glob("*.png"))
```

```python
png_files.sort()

# Verificar Excel
print("\nArchivos Excel:")
for f in excel_files:
    status = "OK" if f.exists() else "FALTA"
    rel_path = f.relative_to(ROOT)
    print(f"  [{status}] {rel_path}")

# Verificar PNGs
print("\nFiguras (PNG):")
expected_pngs = [f"0{i}_" if i < 10 else f"{i}_" for i in range(1, 15)]
for f in png_files:
    rel_path = f.relative_to(ROOT)
    print(f"  [OK] {rel_path}")

print(f"\nTotal de PNG: {len(png_files)} (esperados: 15)")

# Checklist final
print("\n" + "=" * 70)
print("CHECKLIST FINAL:")
print("=" * 70)
checklist = [
    ("entrega_taller_1.xlsx creado", (OUT_EXP / "entrega_taller_1.xlsx").
 ↪exists()),
    ("io_linkages.xlsx existe", (OUT_TAB / "io_linkages.xlsx").exists()),
    ("15 PNG generadas (01-15)", len(png_files) == 15),]

for item, status in checklist:
    mark = " " if status else " "
    print(f"  [{mark}] {item}")

print("=" * 70)
```

```
VERIFICACION FINAL DE ENTREGA
======================================================================

Archivos Excel:
  [OK] outputs/exports/entrega_taller_1.xlsx
  [OK] outputs/tables/io_linkages.xlsx

Figuras (PNG):
  [OK] outputs/figures/01_GDP.png
  [OK] outputs/figures/02_GDP_Var.png
  [OK] outputs/figures/03_consumption.png
  [OK] outputs/figures/04_investment.png
  [OK] outputs/figures/05_unemployment.png
  [OK] outputs/figures/06_inflation.png
```

```
  [OK] outputs/figures/07_fedfunds.png
  [OK] outputs/figures/08_hp_gdp_trend.png
  [OK] outputs/figures/09_hp_inv_trend.png
  [OK] outputs/figures/10_hp_gdp_cycle.png
  [OK] outputs/figures/11_hp_inv_cycle.png
  [OK] outputs/figures/12_hp_gaps_gdp_inv.png
  [OK] outputs/figures/13_hp_gaps_statsmodels.png
  [OK] outputs/figures/14_rolling_corr.png
  [OK] outputs/figures/15_io_scatter_direct.png

Total de PNG: 15 (esperados: 15)

======================================================================
CHECKLIST FINAL:
======================================================================
  [ ] entrega_taller_1.xlsx creado
  [ ] io_linkages.xlsx existe
  [ ] 15 PNG generadas (01-15)
======================================================================
```