Output_Gap_e_Inflacion_Evidencia_para_el_Peru

January 6, 2026

1 Brecha del producto e inflación en el Perú

Autor: Carruitero castillo, david

Línea: Econometría Aplicada

1.1 Resumen

Este trabajo analiza empíricamente la relación entre la brecha del producto y la inflación en la economía peruana, una relación central en la teoría macroeconómica moderna y en los marcos de política monetaria. Dado que el producto potencial no es directamente observable, este se estima mediante el **filtro Hodrick–Prescott** aplicado al Producto Bruto Interno (PBI), este filtro Hodrick–Prescott descompone el Producto Interno Bruto (PIB) en un componente tendencial (crecimiento a largo plazo o PBI potencial) y un componente cíclico (fluctuaciones a corto plazo o brecha del producto. A partir de esta estimación se construye **la brecha del producto, la cual se incorpora como variable explicativa en un modelo econométrico de inflación junto con la inflación rezagada**, capturando la persistencia inflacionaria. Los resultados muestran que una brecha del producto positiva(la economia pera por encima del potencial) se asocia con mayores presiones inflacionarias, en línea con la teoría macroeconómica estándar y la evidencia utilizada por bancos centrales.

1.2 Introducción

La teoría macroeconómica establece que cuando una economía opera por encima de su nivel de producto potencial, se generan presiones inflacionarias como resultado de excesos de demanda agregada.

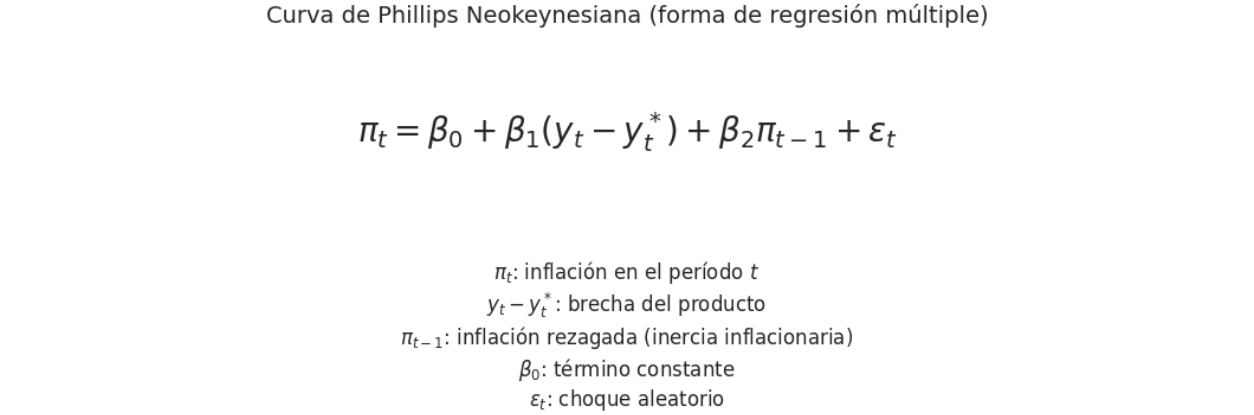
En este contexto, el presente trabajo tiene como objetivo evaluar empíricamente dicha relación para el caso del Perú, utilizando datos macroeconómicos(PBI e IPC) del BCRP y herramientas econométricas estándar(Regresion multiple y filtro Hodrick–Prescott) .

1.3 Marco teórico

El producto potencial se define como el nivel de producción consistente con una inflación estable. La diferencia entre el producto observado y el producto potencial se conoce como brecha del producto y refleja el grado de sobrecalentamiento o holgura de la economía.

En ese sentido, para evaluar la relación entre la brecha del producto y la inflación, se estima un modelo de regresión múltiple dinámica(curva de Phillips Neokeynesiana (CPNK)):

[97]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 4))

# Ecuación principal
plt.text(
    0.5, 0.70,
    r"$\pi_t = \beta_0 + \beta_1 (y_t - y_t^*) + \beta_2 \pi_{t-1} +␣
 ↪\varepsilon_t$",
    fontsize=20,
    ha='center'
)

# Leyenda / explicación de variables
plt.text(
    0.5, 0.38,
    r"$\pi_t$: inflación en el período $t$" "\n"
    r"$y_t - y_t^*$: brecha del producto" "\n"
    r"$\pi_{t-1}$: inflación rezagada (inercia inflacionaria)" "\n"
    r"$\beta_0$: término constante" "\n"
    r"$\varepsilon_t$: choque aleatorio",
    fontsize=12,
    ha='center',
    va='top'
)

plt.axis('off')
plt.title("Curva de Phillips Neokeynesiana (forma de regresión múltiple)",␣
 ↪fontsize=14)
plt.show()
```



1.4 Metodología

Se utilizan series macroeconómicas trimestrales del Perú. El producto potencial se estima mediante el filtro Hodrick–Prescott aplicado al PBI real. La inflación se mide a partir del IPC(indice del precio del consumidor). Ambos indicadores macroecnomicos, se obtienen del BCRP(banco central de reserva del Peru).

#Datos:

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.filters.hp_filter import hpfilter
import statsmodels.api as sm
```

Cargamos los archivos donde se contienen los indicadores: inflacion

[ ]:
```
# cargar, el IPC mensual
ipc = pd.read_excel("IPC MENSUAL.xlsx", parse_dates=["Fecha"])
```

Se carga la serie mensual del Índice de Precios al Consumidor (IPC) desde un archivo Excel y se convierte la variable temporal en un objeto de tipo fecha (datetime)

[ ]:
```
# Convertir la columna 'Fecha' al formato datetime (año-mes), asi trabajamos␣
 ↪una serie de tiempo
ipc["Fecha"] = pd.to_datetime(ipc["Fecha"], format="%Y-%m")

# Establecer la fecha como índice del DataFrame
ipc.set_index("Fecha", inplace=True)

# Verificar el rango temporal de la serie
ipc.index.min(), ipc.index.max()
```

[ ]:
```
(Timestamp('1999-01-01 00:00:00'), Timestamp('2024-12-01 00:00:00'))
```

se transforma el IPC mensual en un IPC trimestral promedio, calculando el promedio de los tres meses que componen cada trimestre

[ ]:
```
# Convertir el IPC mensual a frecuencia trimestral
ipc_trimestral = ipc.resample("Q").mean()

# "resample" se usa para cambiar las frecuencias de una serie de tiempo
```

```
/tmp/ipython-input-1798039718.py:2: FutureWarning: 'Q' is deprecated and will be
removed in a future version, please use 'QE' instead.
  ipc_trimestral = ipc.resample("Q").mean()
```

La inflación se calcula como la variación interanual del IPC trimestral, es decir, el cambio porcentual del índice del trimestre actual respecto al mismo trimestre del año anterior. Esto permite alinear la frecuencia de la inflación con la del PBI trimestral y analizar la relación entre brecha del producto e inflación.

[ ]:
```
ipc_trimestral["inflacion"] = 100 * (
    np.log(ipc_trimestral["IPC"]) -
    np.log(ipc_trimestral["IPC"].shift(4))
)

ipc_trimestral = ipc_trimestral.dropna()

pd.set_option('display.max_rows', None)
ipc_trimestral
```

[ ]:

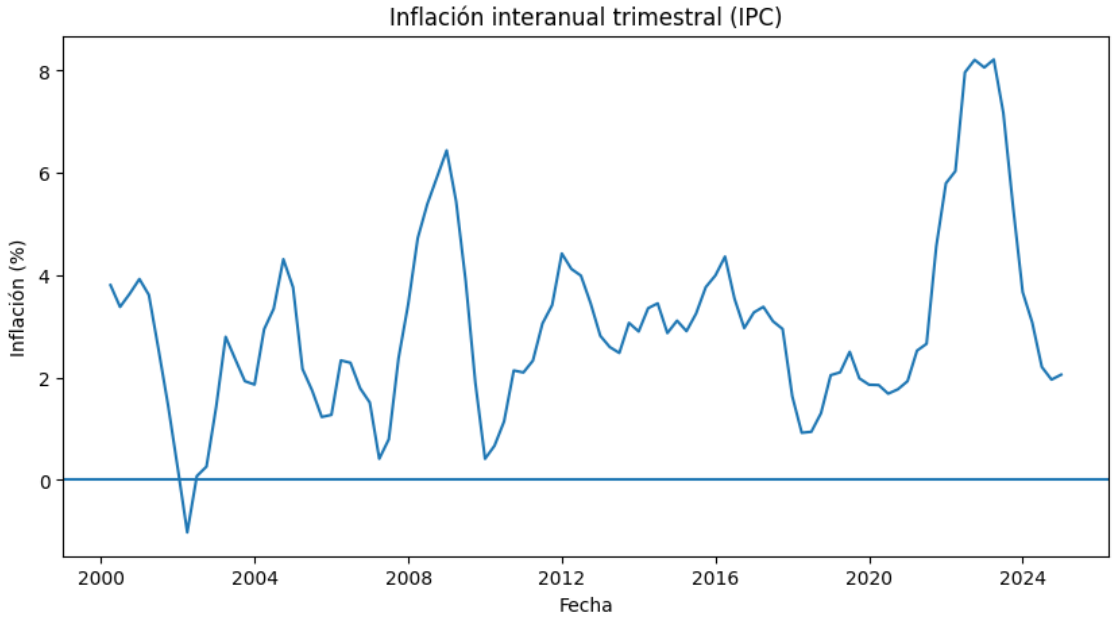
Fecha	IPC	inflacion
2000-03-31	54.953333	3.808100
2000-06-30	55.536667	3.381670
2000-09-30	56.130000	3.640504
2000-12-31	56.610000	3.927018
2001-03-31	56.976667	3.615749
2001-06-30	56.976667	2.559836
2001-09-30	56.953333	1.456179
2001-12-31	56.736667	0.223503
2002-03-31	56.396667	-1.023177
2002-06-30	57.020000	0.076026
2002-09-30	57.103333	0.263027
2002-12-31	57.553333	1.429137
2003-03-31	57.996667	2.797548
2003-06-30	58.383333	2.362838
2003-09-30	58.216667	1.930919
2003-12-31	58.636667	1.864816
2004-03-31	59.730000	2.944887
2004-06-30	60.370000	3.346183
2004-09-30	60.783333	4.314395
2004-12-31	60.886667	3.765400
2005-03-31	61.040000	2.169498
2005-06-30	61.436667	1.751454
2005-09-30	61.536667	1.231757
2005-12-31	61.666667	1.272932
2006-03-31	62.483333	2.337047
2006-06-30	62.860000	2.290320
2006-09-30	62.650000	1.793048
2006-12-31	62.606667	1.512823
2007-03-31	62.743333	0.415248
2007-06-30	63.363333	0.797532
2007-09-30	64.156667	2.376433

2007-12-31	64.793333	3.433095
2008-03-31	65.780000	4.726351
2008-06-30	66.873333	5.391493
2008-09-30	68.066667	5.915961
2008-12-31	69.103333	6.440025
2009-03-31	69.460000	5.443521
2009-06-30	69.523333	3.886214
2009-09-30	69.370000	1.896688
2009-12-31	69.390000	0.413980
2010-03-31	69.926667	0.669603
2010-06-30	70.323333	1.144123
2010-09-30	70.870000	2.139272
2010-12-31	70.863333	2.101037
2011-03-31	71.576667	2.332206
2011-06-30	72.506667	3.057486
2011-09-30	73.333333	3.416804
2011-12-31	74.070000	4.425745
2012-03-31	74.590000	4.123731
2012-06-30	75.460000	3.992420
2012-09-30	75.910000	3.453317
2012-12-31	76.186667	2.817588
2013-03-31	76.553333	2.598122
2013-06-30	77.356667	2.482405
2013-09-30	78.276667	3.070113
2013-12-31	78.430000	2.902004
2014-03-31	79.166667	3.356767
2014-06-30	80.073333	3.451612
2014-09-30	80.556667	2.871131
2014-12-31	80.910000	3.113092
2015-03-31	81.503333	2.908858
2015-06-30	82.720000	3.251853
2015-09-30	83.650000	3.768056
2015-12-31	84.210000	3.997625
2016-03-31	85.140000	4.365304
2016-06-30	85.700000	3.539141
2016-09-30	86.170000	2.968066
2016-12-31	87.013333	3.274768
2017-03-31	88.070000	3.383499
2017-06-30	88.400000	3.101914
2017-09-30	88.750000	2.950134
2017-12-31	88.446667	1.633837
2018-03-31	88.886667	0.923020
2018-06-30	89.236667	0.942005
2018-09-30	89.916667	1.305989
2018-12-31	90.276667	2.047930
2019-03-31	90.776667	2.104013
2019-06-30	91.500000	2.504696

```
2019-09-30   91.720000   1.985714
2019-12-31   91.973333   1.861965
2020-03-31   92.476667   1.855408
2020-06-30   93.056667   1.686965
2020-09-30   93.360000   1.772253
2020-12-31   93.766667   1.931075
2021-03-31   94.843333   2.527005
2021-06-30   95.566667   2.661546
2021-09-30   97.716667   4.560915
2021-12-31   99.363333   5.797374
2022-03-31  100.743333   6.034962
2022-06-30  103.490000   7.965091
2022-09-30  106.076667   8.208997
2022-12-31  107.703333   8.059737
2023-03-31  109.370000   8.216060
2023-06-30  111.186667   7.173548
2023-09-30  111.906667   5.350309
2023-12-31  111.730000   3.670471
2024-03-31  112.786667   3.076150
2024-06-30  113.670000   2.208904
2024-09-30  114.123333   1.961454
2024-12-31  114.053333   2.058093
```

Graficamos la serie de la inflacion

[ ]:
```
#serie de tiempo de la inflacion interanual trimestral

plt.figure(figsize=(10,5))
plt.plot(ipc_trimestral.index, ipc_trimestral["inflacion"])
plt.axhline(0)  # línea de referencia
plt.title("Inflación interanual trimestral (IPC)")
plt.xlabel("Fecha")
plt.ylabel("Inflación (%)")
plt.show()
```



Cargamos los archivos donde se contienen los indicadores: PBI

[ ]:
```
#  archivo
pbi = pd.read_excel("PBI.xlsx")

#  Convertir Fecha correctamente (YYYYTQ)
pbi["Fecha"] = pbi["Fecha"].astype(str)

periodos = pd.PeriodIndex(
    pbi["Fecha"].str.replace("T", "Q"),
    freq="Q"
)


pbi.index = periodos.to_timestamp(how="end")
pbi.index = pbi.index.normalize()



#  Eliminar columna Fecha
pbi.drop(columns="Fecha", inplace=True)

#  Verificación
print(pbi.index.min(), pbi.index.max())
print(pbi.tail())
```

```
2000-03-31 00:00:00 2024-12-31 00:00:00
```

```
                PBI
Fecha
2023-12-31  148898
2024-03-31  135394
2024-06-30  145965
2024-09-30  148689
2024-12-31  155552
```

Se usa log(PBI) para que el ciclo (output gap) represente desviaciones porcentuales respecto al PBI potencial y la serie sea más estable y comparable con la inflación.

[ ]:
```
#  Logaritmo del PBI
pbi["log_pbi"] = np.log(pbi["PBI"])
```

[ ]:
```
pbi.index.min(), pbi.index.max()
```

[ ]:
```
(Timestamp('2000-03-31 00:00:00'), Timestamp('2024-12-31 00:00:00'))
```

[ ]:
```
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)

pbi
```

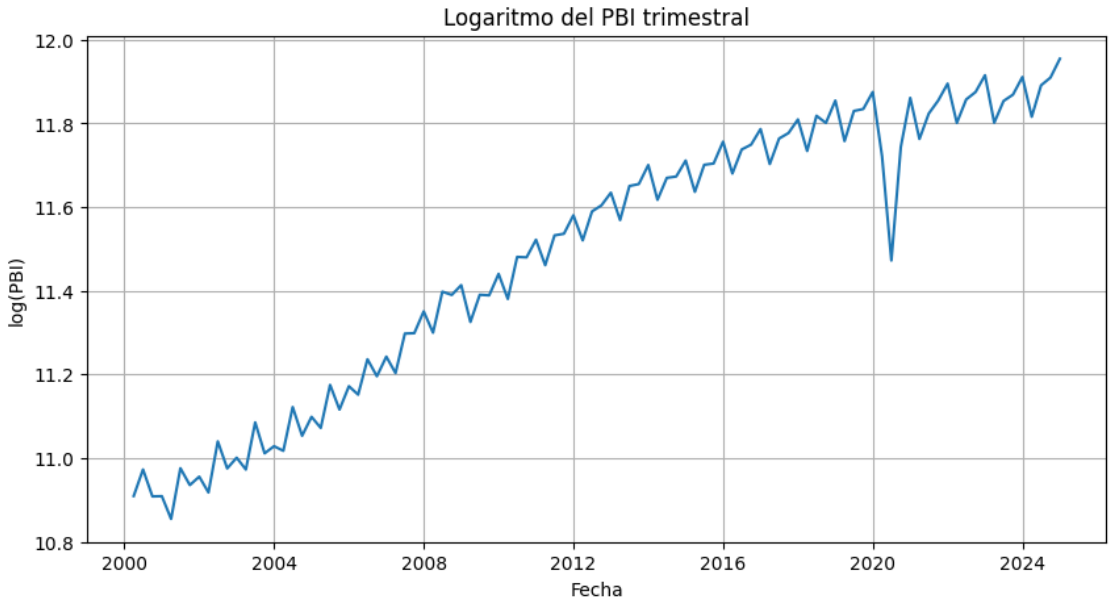
[ ]:
```
                PBI    log_pbi
Fecha
2000-03-31   54675  10.909162
2000-06-30   58256  10.972602
2000-09-30   54622  10.908192
2000-12-31   54655  10.908796
2001-03-31   51760  10.854373
2001-06-30   58431  10.975602
2001-09-30   56120  10.935248
2001-12-31   57268  10.955497
2002-03-31   55138  10.917594
2002-06-30   62307  11.039829
2002-09-30   58404  10.975140
2002-12-31   59924  11.000832
2003-03-31   58249  10.972482
2003-06-30   65202  11.085245
2003-09-30   60552  11.011258
2003-12-31   61589  11.028239
2004-03-31   60914  11.017218
2004-06-30   67640  11.121955
2004-09-30   63146  11.053205
2004-12-31   66071  11.098485
2005-03-31   64341  11.071952
2005-06-30   71310  11.174792
2005-09-30   67230  11.115875
```

2005-12-31	71090	11.171702
2006-03-31	69671	11.151539
2006-06-30	75824	11.236170
2006-09-30	72806	11.195554
2006-12-31	76297	11.242389
2007-03-31	73354	11.203052
2007-06-30	80626	11.297576
2007-09-30	80700	11.298494
2007-12-31	85013	11.350559
2008-03-31	80796	11.299683
2008-06-30	89118	11.397717
2008-09-30	88430	11.389967
2008-12-31	90526	11.413392
2009-03-31	82892	11.325294
2009-06-30	88464	11.390351
2009-09-30	88341	11.388960
2009-12-31	92995	11.440301
2010-03-31	87579	11.380297
2010-06-30	96844	11.480857
2010-09-30	96742	11.479803
2010-12-31	100905	11.521935
2011-03-31	94948	11.461085
2011-06-30	101962	11.532355
2011-09-30	102364	11.536290
2011-12-31	106973	11.580332
2012-03-31	100749	11.520388
2012-06-30	107972	11.589627
2012-09-30	109551	11.604145
2012-12-31	112918	11.634417
2013-03-31	105765	11.568975
2013-06-30	114735	11.650380
2013-09-30	115277	11.655093
2013-12-31	120646	11.700616
2014-03-31	111006	11.617340
2014-06-30	116983	11.669784
2014-09-30	117378	11.673155
2014-12-31	121935	11.711243
2015-03-31	113149	11.636461
2015-06-30	120699	11.701055
2015-09-30	121082	11.704223
2015-12-31	127565	11.756381
2016-03-31	118217	11.680277
2016-06-30	125207	11.737724
2016-09-30	126667	11.749317
2016-12-31	131481	11.786618
2017-03-31	120918	11.702868
2017-06-30	128523	11.763863

```
2017-09-30  130232  11.777073
2017-12-31  134533  11.809565
2018-03-31  124747  11.734043
2018-06-30  135716  11.818320
2018-09-30  133408  11.801167
2018-12-31  140746  11.854712
2019-03-31  127741  11.757760
2019-06-30  137278  11.829763
2019-09-30  137930  11.834502
2019-12-31  143645  11.875100
2020-03-31  123176  11.721370
2020-06-30   96022  11.472333
2020-09-30  125946  11.743609
2020-12-31  141664  11.861213
2021-03-31  128389  11.762820
2021-06-30  136438  11.823626
2021-09-30  140755  11.854776
2021-12-31  146562  11.895204
2022-03-31  133416  11.801227
2022-06-30  141105  11.857260
2022-09-30  143649  11.875128
2022-12-31  149539  11.915313
2023-03-31  133469  11.801625
2023-06-30  140603  11.853696
2023-09-30  142750  11.868850
2023-12-31  148898  11.911017
2024-03-31  135394  11.815944
2024-06-30  145965  11.891122
2024-09-30  148689  11.909612
2024-12-31  155552  11.954735
```

Graficamos la serie del PBI

[ ]:
```
plt.figure(figsize=(10, 5))
plt.plot(pbi.index, pbi["log_pbi"])
plt.title("Logaritmo del PBI trimestral")
plt.xlabel("Fecha")
plt.ylabel("log(PBI)")
plt.grid(True)
plt.show()
```



#filtro de hodrick-prescott

Filtrar HP para separar PBI en tendencia (potencial) y ciclo (output gap). Este ultimo servira para nuestro modelo econometrico simple

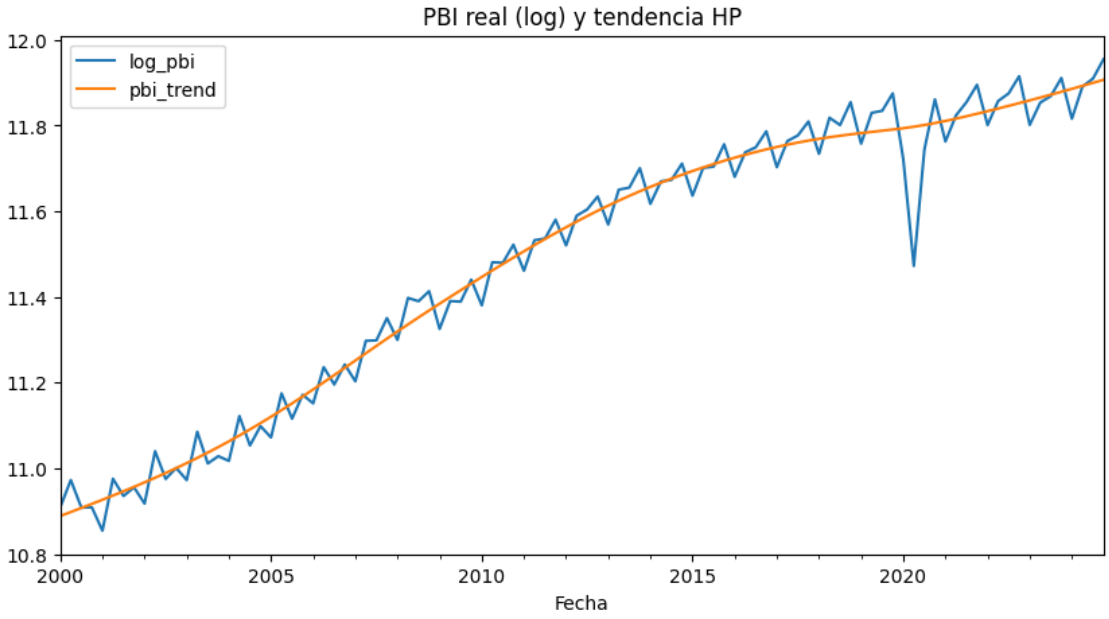
- PBI tendencial (o potencial): muestra el nivel "normal" de la economía, es decir, cómo crecería sin los altibajos temporales. Piensa en una línea suave que refleja la capacidad estable de la economía.
- PBI cíclico (o output gap): muestra las desviaciones respecto a esa tendencia. Si está por encima de la línea tendencial, la economía está "caliente" (expansión); si está por debajo, está "fría" (recesión o desaceleración).

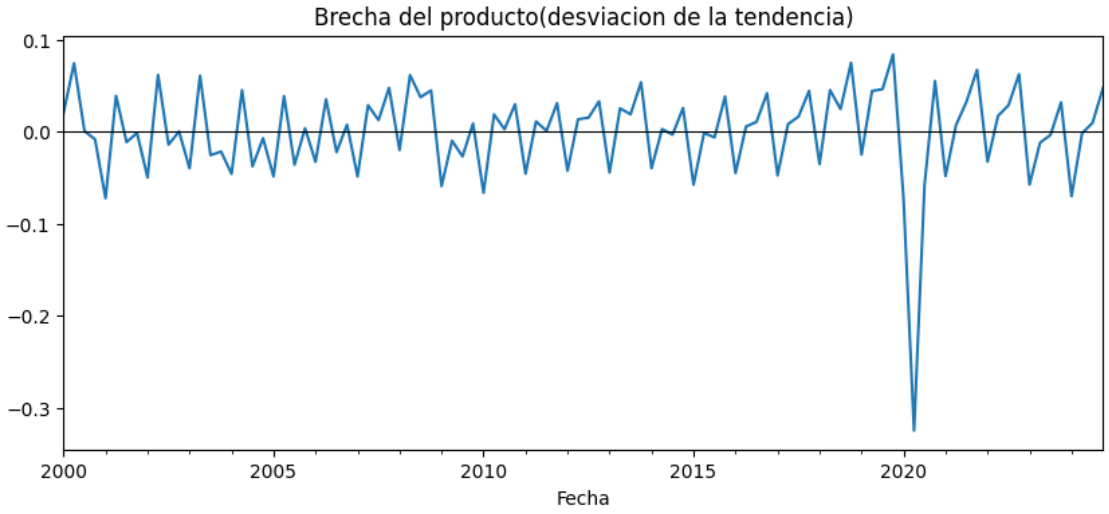
En otras palabras, el ciclo indica los movimientos a corto plazo, mientras que la tendencia muestra el crecimiento de largo plazo.

[ ]:
```
from statsmodels.tsa.filters.hp_filter import hpfilter

# Filtro HP
cycle, trend = hpfilter(pbi["log_pbi"], lamb=1600)

#  resultados guardados
pbi["pbi_trend"] = trend
pbi["pbi_cycle"] = cycle
```

[ ]:
```
import matplotlib.pyplot as plt

pbi[["log_pbi", "pbi_trend"]].plot(figsize=(10,5))
plt.title("PBI real (log) y tendencia HP")
```

```
plt.show()

pbi["pbi_cycle"].plot(figsize=(10,4))
plt.axhline(0, color="black", linewidth=0.8)
plt.title("Brecha del producto(desviacion de la tendencia)")
plt.show()
```





2 Hipotesis

Se plantea la hipótesis de que la brecha del producto (output gap) tiene un efecto positivo sobre la inflación: a mayor expansión económica por encima de la tendencia, mayor será la inflación. Esta relación se tratará de verificar mediante la estimación del modelo econométrico.

3 Planteamiento del modelo econometrico

Primero se procede a unir todos los datos que necesitaremos para nuestro modelo de regresion multiple: ciclo del PBI, inflacion y inflacion rezagada

[ ]:
```
df = pbi[["pbi_cycle"]].join(
    ipc_trimestral[["inflacion"]],
    how="inner"
)
df["inflacion_lag"] = df["inflacion"].shift(1)
df = df.dropna()
df.head()
```

[ ]:

Fecha	pbi_cycle	inflacion	inflacion_lag
2000-06-30	0.074610	3.381670	3.808100
2000-09-30	0.000775	3.640504	3.381670
2000-12-31	-0.008118	3.927018	3.640504
2001-03-31	-0.072170	3.615749	3.927018
2001-06-30	0.039243	2.559836	3.615749

3.1 Estimacion del modelo econometrico

[ ]:
```
import statsmodels.api as sm

X = df[["pbi_cycle", "inflacion_lag"]]
X = sm.add_constant(X)

y = df["inflacion"]

modelo = sm.OLS(y, X).fit()

modelo.summary()
```

[ ]:

Dep. Variable:	inflacion	**R-squared:**	0.815
Model:	OLS	**Adj. R-squared:**	0.811
Method:	Least Squares	**F-statistic:**	210.8
Date:	Tue, 06 Jan 2026	**Prob (F-statistic):**	7.48e-36
Time:	02:57:31	**Log-Likelihood:**	-114.35
No. Observations:	99	**AIC:**	234.7
Df Residuals:	96	**BIC:**	242.5
Df Model:	2		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
const	0.2914	0.154	1.896	0.061	-0.014	0.596
pbi_cycle	2.2582	1.547	1.460	0.148	-0.812	5.329
inflacion_lag	0.8972	0.044	20.380	0.000	0.810	0.985

Omnibus:	0.302	**Durbin-Watson:**	0.996
Prob(Omnibus):	0.860	**Jarque-Bera (JB):**	0.049
Skew:	0.008	**Prob(JB):**	0.976
Kurtosis:	3.107	**Cond. No.**	71.0

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Dado que los residuos presentan autocorrelación(DW=0.996), se estimó el modelo utilizando errores estándar robustos de Newey–West, adecuados para datos trimestrales, garantizando inferencia estadística válida.

[ ]:
```
modelo_nw = sm.OLS(y, X).fit(
    cov_type='HAC',
    cov_kwds={'maxlags': 4}
)
modelo_nw.summary()
```

[ ]:

Dep. Variable:	inflacion	**R-squared:**	0.815
Model:	OLS	**Adj. R-squared:**	0.811
Method:	Least Squares	**F-statistic:**	125.0
Date:	Tue, 06 Jan 2026	**Prob (F-statistic):**	1.87e-27
Time:	02:37:38	**Log-Likelihood:**	-114.35
No. Observations:	99	**AIC:**	234.7
Df Residuals:	96	**BIC:**	242.5
Df Model:	2		
Covariance Type:	HAC		

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
const	0.2914	0.155	1.882	0.060	-0.012	0.595
pbi_cycle	2.2582	1.325	1.704	0.088	-0.339	4.856
inflacion_lag	0.8972	0.057	15.646	0.000	0.785	1.010

Omnibus:	0.302	**Durbin-Watson:**	0.996
Prob(Omnibus):	0.860	**Jarque-Bera (JB):**	0.049
Skew:	0.008	**Prob(JB):**	0.976
Kurtosis:	3.107	**Cond. No.**	71.0

Notes:
[1] Standard Errors are heteroscedasticity and autocorrelation robust (HAC) using 4 lags and without small sample correction

Los resultados evidencian una elevada persistencia inflacionaria (coef. rezago = 0.897, (p<0.01)), la cual, lejos de indicar descontrol, refleja la consolidación de la estabilidad de precios en torno al objetivo del 2%. Esta dinámica es consistente con un anclaje sólido de las expectativas dentro del esquema de Metas de Inflación del BCRP. Asimismo, se halló un impacto positivo y significativo de la brecha del producto (coef. = 2.258, (p=0.088)), confirmando que la política monetaria mantiene capacidad de respuesta ante presiones de demanda en un contexto de predictibilidad macroeconómica.

[ ]:
```
# Obtener los coeficientes redondeados
const = modelo_nw.params['const']
pbi = modelo_nw.params['pbi_cycle']
lag = modelo_nw.params['inflacion_lag']

# Mostrar la ecuación de la regresión
print("Ecuación de la regresión múltiple:")
print(f"inflacion = {const:.4f} + {pbi:.4f} × pbi_cycle + {lag:.4f} ×␣
 ↪inflacion_lag")
print()
```

```
Ecuación de la regresión múltiple:
inflacion = 0.2914 + 2.2582 × pbi_cycle + 0.8972 × inflacion_lag
```

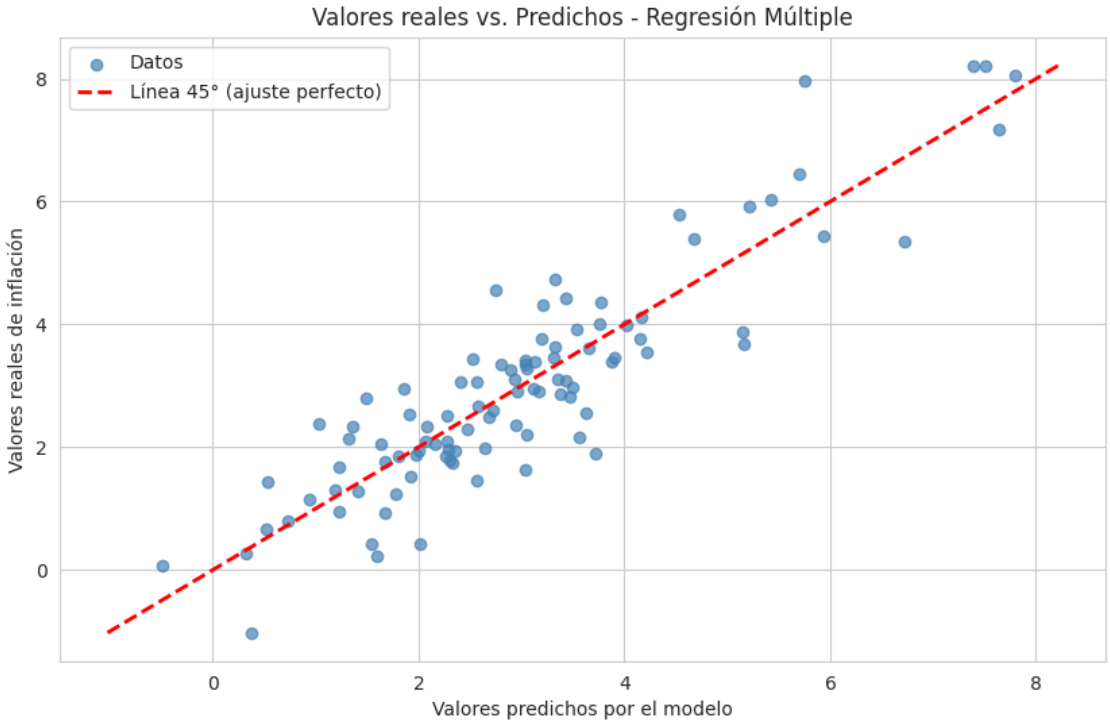
3.2 Rendimiento del modelo

Permite visualizar el rendimiento de un modelo de regresión. El eje x representa los valores reales y el eje y los valores predichos. Idealmente, si las predicciones son correctas, los puntos se ubicarán a lo largo de una línea recta con una pendiente de 1.

De lo que se oberva de los resultados podemos decir, que el modelo tiene un buen poder predictivo. Cuanto más pegados estén los puntos a la línea roja, mejor es el modelo

[ ]:
```
# Gráfica: Valores reales vs. Predichos
predichos = modelo_nw.predict(X)
plt.figure(figsize=(10, 6))
sns.set_style("whitegrid")
plt.scatter(predichos, y, alpha=0.7, color='steelblue', label='Datos')
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--', lw=2, label='Línea 45°␣
 ↪(ajuste perfecto)')
plt.xlabel('Valores predichos por el modelo')
plt.ylabel('Valores reales de inflación')
```

```
plt.title('Valores reales vs. Predichos - Regresión Múltiple')
plt.legend()
plt.show()
```



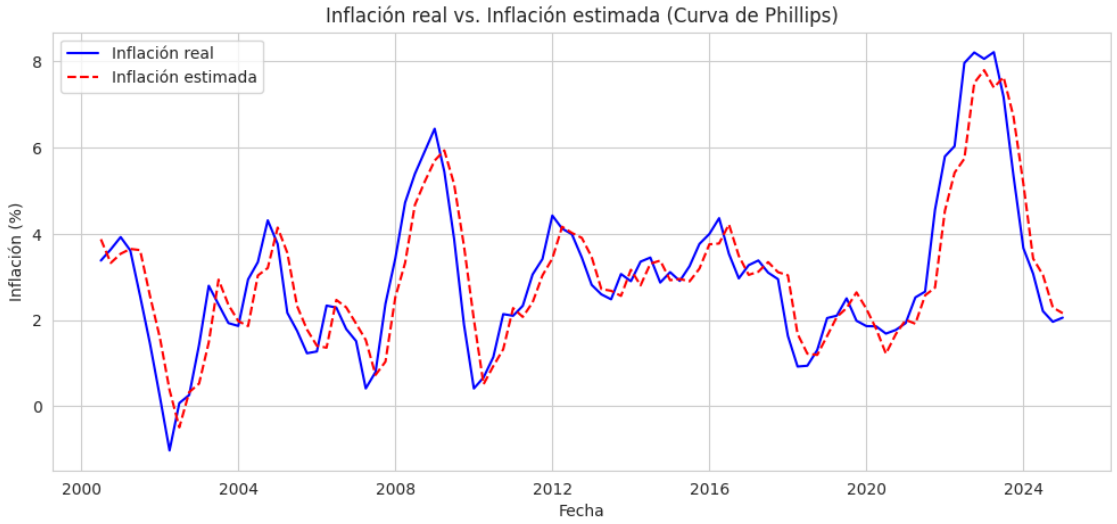
3.3 Gráfico comparativo de inflación real vs. inflación estimada

Se visualiza cómo el modelo captura los movimientos de la inflación a lo largo del tiempo, permitiendo evaluar el ajuste del modelo.

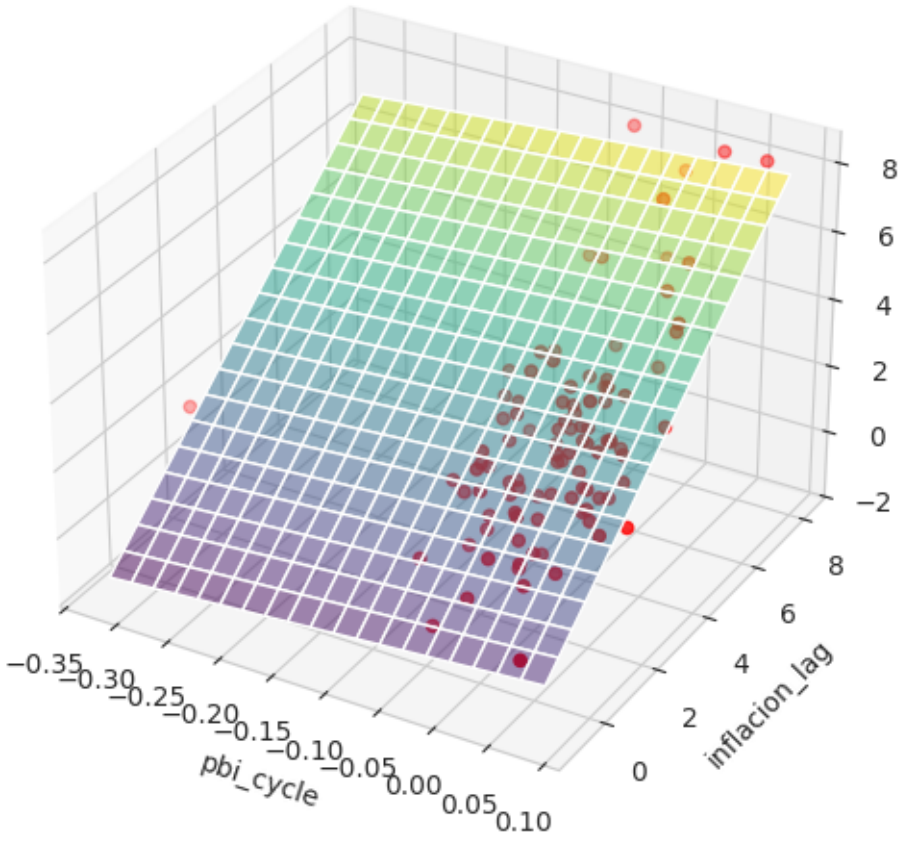
Ademas, podemos decir que la gráfica muestra que el modelo de regresión múltiple, basado en el ciclo del PBI y la inflación rezagada, captura con gran precisión la evolución temporal de la inflación real entre 2000 y 2024.

[ ]:
```
import matplotlib.pyplot as plt

# Inflación predicha
df['inflacion_pred'] = modelo_nw.predict(X)

plt.figure(figsize=(12,5))
plt.plot(df.index, df['inflacion'], label="Inflación real", color="blue")
plt.plot(df.index, df['inflacion_pred'], label="Inflación estimada",␣
 ↪color="red", linestyle='--')
plt.title("Inflación real vs. Inflación estimada (Curva de Phillips)")
plt.xlabel("Fecha")
```

```
plt.ylabel("Inflación (%)")
plt.legend()
plt.show()
```



[ ]:
```
from mpl_toolkits.mplot3d import Axes3D

X1, X2 = np.meshgrid(
    np.linspace(df['pbi_cycle'].min(), df['pbi_cycle'].max(), 20),
    np.linspace(df['inflacion_lag'].min(), df['inflacion_lag'].max(), 20)
)

Y = modelo.params['const'] + modelo.params['pbi_cycle']*X1 + modelo.
 ↪params['inflacion_lag']*X2

fig = plt.figure(figsize=(10,6))
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X1, X2, Y, alpha=0.5, cmap='viridis')
ax.scatter(df['pbi_cycle'], df['inflacion_lag'], df['inflacion'], color='red')
ax.set_xlabel("pbi_cycle")
ax.set_ylabel("inflacion_lag")
ax.set_zlabel("Inflación real")
plt.show()
```



El gráfico en tres dimensiones muestra cómo la inflación depende al mismo tiempo del ciclo económico y de su comportamiento pasado. La superficie representa la inflación estimada por el modelo, mientras que los puntos corresponden a los datos observados. Se aprecia que la inflación tiende a ser mayor cuando la economía está en expansión y cuando la inflación previa es elevada.

4 CONCLUSION:

El análisis confirma que la inflación en Perú está influenciada por su valor pasado y por la actividad económica. La brecha del producto positiva (expansión) tiende a asociarse con mayor inflación, mientras que la inflación rezagada muestra persistencia significativa. En conjunto, el modelo respalda parcialmente la Curva de Phillips y ofrece evidencia cuantitativa sobre la relación entre ciclo económico e inflación.