Estimación del multiplicador fiscal en México

Max Lugo Delgadillo

No citar, trabajo en proceso.

```
#Código Multiplicador Fiscal

mport pandas

import numpy

#series de datos trimestral

fecha = pandas.date_range('1993Q1', periods=92, freq='Q')

...Data = pandas.read_excel("BaseAgregada2.xlsx") #importar de excel

...# la series originales son desestacionalizadas a pesos constantes

# por X-13arima-Seats en millones se generar las series en per cápita y logaritmos

# se declaran como series y luego se les pone un índice de tiempo

...Y = pandas.Series( numpy.log( ( Data["PIB real"] ) ) )

G = pandas.Series( numpy.log( ( Data["Gasto público neto pagado real"] ) ) )

GK = pandas.Series( numpy.log( ( Data["Gasto de capital presupuestario real"]  ) ) )

GC = pandas.Series( numpy.log( ( Data["Gasto corriente real"]  ) ) )

CC = pandas.Series( ( ( Data["Cuenta corriente real"]  ) ) )

IT = pandas.Series( numpy.log( ( Data["Ingreso Tributario más aportaciones real"]  ) ) )

GS = pandas.Series( numpy.log( ( Data["Gasto real sin costo financiero ni adefas"]  ) ) )

GCS = pandas.Series( numpy.log( ( Data["Gasto corriente real sin costo financiero ni adefas"]  ) ) )

Con = pandas.Series( numpy.log( ( Data["Consumo real"]  ) ) )

IP = pandas.Series( numpy.log( ( Data["Ingresos públicos presupuestarios"]  ) ) )

I = pandas.Series( numpy.log( ( Data["Inversión real"]  ) ) )

...Y_d = pandas.Series( numpy.log( ( Data["PIB real d"] ) ) )

G_d = pandas.Series( numpy.log( ( Data["Gasto público neto pagado real d"] ) ) )

GK_d = pandas.Series( numpy.log( ( Data["Gasto de capital presupuestario real d"]  ) ) )

GC_d = pandas.Series( numpy.log( ( Data["Gasto corriente real d"]  ) ) )
```

```
CC_d = pandas.Series( numpy.log( ( Data["Cuenta corriente real d"]  ) ) )

IT_d = pandas.Series( numpy.log( ( Data["Ingreso Tributario más aportaciones real d"]  ) ) )

GS_d = pandas.Series( numpy.log( ( Data["Gasto real sin costo financiero ni adefas d"]  ) ) )

GCS_d = pandas.Series( numpy.log( ( Data["Gasto corriente real sin costo financiero ni adefas d"]  ) ) )

Con_d = pandas.Series( numpy.log( ( Data["Consumo real d"]  ) ) )

IP_d = pandas.Series( numpy.log( ( Data["Ingresos públicos presupuestarios d"]  ) ) )

I_d = pandas.Series( numpy.log( ( Data["Inversión real d"]  ) ) )

...C95 = pandas.Series(Data["Crisis95"])

C08 = pandas.Series(Data["Crisis08"])

PCF = pandas.Series(Data["PCF"])

EZP = pandas.Series(Data["EZP"])

VFQ = pandas.Series(Data["VFQ"])

FCH = pandas.Series(Data["FCH"])

...ir = pandas.Series(Data["Tasa de interés real"])

xr = pandas.Series(Data["Tipo de cambio real"])

inf = pandas.Series(Data["Inflación"])

inf_d = pandas.Series( ( Data["inflación d"]  ) )

...# índices de tiempo

Y.index = pandas.DatetimeIndex(fecha)

G.index = pandas.DatetimeIndex(fecha)

GK.index = pandas.DatetimeIndex(fecha)

GC.index = pandas.DatetimeIndex(fecha)

CC.index = pandas.DatetimeIndex(fecha)
```

```
IT.index = pandas.DatetimeIndex(fecha)

GS.index = pandas.DatetimeIndex(fecha)

GCS.index = pandas.DatetimeIndex(fecha)

Con.index = pandas.DatetimeIndex(fecha)

I.index = pandas.DatetimeIndex(fecha)

...Y_d.index = pandas.DatetimeIndex(fecha)

G_d.index = pandas.DatetimeIndex(fecha)

GK_d.index = pandas.DatetimeIndex(fecha)

GC_d.index = pandas.DatetimeIndex(fecha)

CC_d.index = pandas.DatetimeIndex(fecha)

IT_d.index = pandas.DatetimeIndex(fecha)

GS_d.index = pandas.DatetimeIndex(fecha)

GCS_d.index = pandas.DatetimeIndex(fecha)

Con_d.index = pandas.DatetimeIndex(fecha)

I_d.index = pandas.DatetimeIndex(fecha)

...# la series originales son desestacionalizadas por X-13arima-Seats en millones se generar las series en per cápita y logaritmos

y = pandas.Series( numpy.log( ( Data["PIB real d"] / Data["Pob"] )*1000000 ) , name='PIB per cápita d')

g = pandas.Series(numpy.log( ( Data["Gasto público neto pagado real d"] / Data["Pob"] )*1000000 ), name='G. per cápita d' )

gk = pandas.Series(numpy.log( ( Data["Gasto de capital presupuestario real d"] / Data["Pob"] )*1000000), name='G. capital per cápita d'
)

gc = pandas.Series(numpy.log( ( Data["Gasto corriente real d"] / Data["Pob"] )*1000000), name='G. corriente per cápita d' )

cc = pandas.Series(( ( Data["Cuenta corriente real d"] / Data["Pob"] )*1000000), name='Cuenta corriente per cápita d' )

it = pandas.Series(numpy.log( ( Data["Ingreso Tributario más aportaciones real d"] / Data["Pob"] )*1000000),

        name='Ingreso tributario más aportaciones per cápita d' )
```

```
gs = pandas.Series(numpy.log( ( Data["Gasto real sin costo financiero ni adefas d"] / Data["Pob"] )*1000000),
                   name='Gasto real sin costo financiero ni adefas per cápita d' )
gcs = pandas.Series(numpy.log( ( Data["Gasto corriente real sin costo financiero ni adefas d"] / Data["Pob"] )*1000000),
                    name='Gasto corriente real sin costo financiero ni adefas per cápita d' )
con = pandas.Series(numpy.log( ( Data["Consumo real d"] / Data["Pob"] )*1000000),name='Consumo per cápita d' )
ip = pandas.Series(numpy.log( ( Data["Ingresos públicos presupuestarios d"] / Data["Pob"] )*1000000),
                   name='Ingresos públicos presupuestarios per cápita d' )
i = pandas.Series(numpy.log( ( Data["Inversión real d"] / Data["Pob"] )*1000000),
                  name='Inversión per cápita d' )
...y.index = pandas.DatetimeIndex(fecha)
g.index = pandas.DatetimeIndex(fecha)
gk.index = pandas.DatetimeIndex(fecha)
gc.index = pandas.DatetimeIndex(fecha)
cc.index = pandas.DatetimeIndex(fecha)
it.index = pandas.DatetimeIndex(fecha)
gs.index = pandas.DatetimeIndex(fecha)
gcs.index = pandas.DatetimeIndex(fecha)
con.index = pandas.DatetimeIndex(fecha)
ip.index = pandas.DatetimeIndex(fecha)
i.index = pandas.DatetimeIndex(fecha)
C95.index = pandas.DatetimeIndex(fecha)
C08.index = pandas.DatetimeIndex(fecha)
...PCF.index = pandas.DatetimeIndex(fecha)
EZP.index = pandas.DatetimeIndex(fecha)
```

```
VFQ.index = pandas.DatetimeIndex(fecha)

FCH.index = pandas.DatetimeIndex(fecha)

...ir.index = pandas.DatetimeIndex(fecha)

xr.index = pandas.DatetimeIndex(fecha)

inf.index = pandas.DatetimeIndex(fecha)

inf_d.index = pandas.DatetimeIndex(fecha)

...get_ipython().magic('matplotlib inline')

import matplotlib.pyplot as plt

...fig = plt.figure(figsize=(40, 32))

...plt.subplot(431)

plt.plot(Y, 'b-', label='Serie original' )

plt.plot(Y_d, 'r-', label='Serie desestacionalizada' )

plt.legend( loc='upper left' )

plt.title('PIB',fontsize=25)

plt.grid(True)

...plt.subplot(432)

plt.plot(G, 'b-', label='Serie original')

plt.plot(G_d, 'r-', label='Serie desestacionalizada' )

plt.legend( loc='upper left' )

plt.title('Gasto público neto pagado',fontsize=25)

plt.grid(True)

...plt.subplot(433)

plt.plot(GS, 'b-', label='Serie original')
```

```
plt.plot(GS_d, 'r-', label='Serie desestacionalizada' )

plt.legend( loc='upper left' )

plt.title('Gasto sin costo financiero ni adefas',fontsize=25)

plt.grid(True)

...plt.subplot(434)

plt.plot(GK, 'b-', label='Serie original')

plt.plot( GK_d, 'r-', label='Serie desestacionalizada')

plt.legend( loc='upper left' )

plt.title('Gasto de capital presupuestario',fontsize=25)

plt.grid(True)

...#plt.plot(GC, 'b-' )

#plt.plot(GC_d, 'r-')

#plt.legend( loc='upper left' )

#plt.grid(True)

...plt.subplot(435)

plt.plot(GCS, 'b-', label='Serie original')

plt.plot(GCS_d, 'r-', label='Serie desestacionalizada')

plt.legend( loc='upper left' )

plt.title('Gasto corriente sin costo financiero ni adefas',fontsize=25)

plt.grid(True)

...plt.subplot(436)

plt.plot(IT, 'b-', label='Serie original' )

plt.plot(IT_d, 'r-', label='Serie desestacionalizada')
```

```
plt.legend( loc='upper left' )

plt.title('Ingreso tributario más aportaciones',fontsize=25)

plt.grid(True)

...plt.subplot(437)

plt.plot(CC, 'b-', label='Serie original'  )

plt.legend( loc='upper left' )

plt.title('Balanza comercial',fontsize=25)

plt.grid(True)

...plt.subplot(438)

plt.plot(Con, 'b-', label='Serie original' )

plt.plot(Con_d, 'r-', label='Serie desestacionalizada')

plt.legend( loc='upper left' )

plt.title('Consumo',fontsize=25)

plt.grid(True)

...plt.subplot(439)

plt.plot(I, 'b-',label='Serie original' )

plt.plot(I_d, 'r-', label='Serie desestacionalizada')

plt.legend( loc='upper left' )

plt.title('Inversión',fontsize=25)

plt.grid(True)

...plt.subplot(4,3,10)

plt.plot(ir, 'b-',label='Serie original' )

plt.legend( loc='upper left' )

plt.title('Tasa de interés',fontsize=25)
```

```
plt.grid(True)

...plt.subplot(4,3,11)

plt.plot(xr, 'b-',label='Serie original' )

plt.legend( loc='upper left' )

plt.title('Tipo de cambio',fontsize=25)

plt.grid(True)

...plt.subplot(4,3,12)

plt.plot(inf, 'b-',label='Serie original' )

plt.plot(inf_d, 'r-', label='Serie desestacionalizada' )

plt.legend( loc='upper left' )

plt.title(inf.name,fontsize=25)

plt.grid(True)

...#plt.plot(MF_puntual2[:,2], 'r-',label='Estimación puntual' )

#plt.legend( loc='lower left' )

#plt.title('Choque al gasto corriente')

...plt.show()

...fig.savefig('Series_desetacionalizadas.eps', format='eps', dpi=10000)

...# tendencias y ciclo con filtro HP

import statsmodels.api

...y_cycle, y_trend = statsmodels.api.tsa.filters.hpfilter(y,1600)

g_cycle, g_trend = statsmodels.api.tsa.filters.hpfilter(g,1600)

gk_cycle, gk_trend = statsmodels.api.tsa.filters.hpfilter(gk,1600)

gc_cycle, gc_trend = statsmodels.api.tsa.filters.hpfilter(gc,1600)
```

```
cc_cycle, cc_trend = statsmodels.api.tsa.filters.hpfilter(cc,1600)

it_cycle, it_trend = statsmodels.api.tsa.filters.hpfilter(it,1600)

xr_cycle, xr_trend = statsmodels.api.tsa.filters.hpfilter(xr,1600)

ir_cycle, ir_trend = statsmodels.api.tsa.filters.hpfilter(ir,1600)

inf_cycle, inf_trend = statsmodels.api.tsa.filters.hpfilter(inf_d,1600)

gs_cycle, gs_trend = statsmodels.api.tsa.filters.hpfilter(gs,1600)

gcs_cycle, gcs_trend = statsmodels.api.tsa.filters.hpfilter(gcs,1600)

con_cycle, con_trend = statsmodels.api.tsa.filters.hpfilter(con,1600)

ip_cycle, ip_trend = statsmodels.api.tsa.filters.hpfilter(ip,1600)

i_cycle, i_trend = statsmodels.api.tsa.filters.hpfilter(i,1600)

...y_cycle =  pandas.Series(y_cycle, name='PIB per cápita (ciclo)')

g_cycle =  pandas.Series(g_cycle, name='G. per cápita (ciclo)')

gk_cycle =  pandas.Series(gk_cycle, name='G. capital per cápita (ciclo)')

gc_cycle =  pandas.Series(gc_cycle, name='G. corriente per cápita (ciclo)')

cc_cycle =  pandas.Series(cc_cycle, name='Cuenta corriente per cápita (ciclo)')

it_cycle =  pandas.Series(it_cycle, name='Ingreso tributario per cápita (ciclo)')

gs_cycle =  pandas.Series(gs_cycle, name='Gasto real sin costo financiero ni adefas per cápita (ciclo)')

gcs_cycle =  pandas.Series(gcs_cycle, name='Gasto corriente real sin costo financiero ni adefas per cápita (ciclo)')

con_cycle =  pandas.Series(con_cycle, name='Consumo per cápita (ciclo)')

xr_cycle = pandas.Series(xr_cycle, name='Tipo de cambio real (ciclo)')

ir_cycle = pandas.Series(ir_cycle, name='Tasa de interés real (ciclo)')

ip_cycle = pandas.Series(ip_cycle, name='Ingresos públicos presupuestarios per cápita (ciclo)')

i_cycle = pandas.Series(i_cycle, name='Inversión real per cápita (ciclo)')
```

...y_trend = pandas.Series(y_trend, name='PIB per cápita (tendencia)')

g_trend = pandas.Series(g_trend, name='G. per cápita (tendencia)')

gk_trend = pandas.Series(gk_trend, name='G. capital per cápita (tendencia)')

gc_trend = pandas.Series(gc_trend, name='G. corriente per cápita (tendencia)')

cc_trend = pandas.Series(cc_trend, name='Cuenta corriente per cápita (tendencia)')

it_trend = pandas.Series(it_trend, name='Ingreso tributario per cápita (tendencia)')

gs_trend = pandas.Series(gs_trend, name='Gasto real sin costo financiero ni adefas per cápita (tendencia)')

gcs_trend = pandas.Series(gcs_trend, name='Gasto corriente real sin costo financiero ni adefas per cápita (tendencia)')

con_trend = pandas.Series(con_trend, name='Consumo per cápita (tendencia)')

xr_trend = pandas.Series(xr_trend, name='Tipo de cambio real (tendencia)')

ir_trend = pandas.Series(ir_trend, name='Tasa de interés real (tendencia)')

ip_trend = pandas.Series(ip_trend, name='Ingresos públicos presupuestarios per cápita (tendencia)')

i_trend = pandas.Series(i_trend, name='Inversión real per cápita (tendencia)')

...statsmodels.tsa.stattools.adfuller(y_cycle, maxlag=10, regression='c', autolag='AIC', store=False, regresults=True)

...statsmodels.tsa.stattools.adfuller(g_cycle, maxlag=10, regression='c', autolag='AIC', store=False, regresults=False)

...statsmodels.tsa.stattools.adfuller(gs_cycle, maxlag=10, regression='c', autolag='AIC', store=False, regresults=False)

...statsmodels.tsa.stattools.adfuller(gcs_cycle, maxlag=10, regression='c', autolag='AIC', store=False, regresults=False)

...statsmodels.tsa.stattools.adfuller(gk_cycle, maxlag=10, regression='c', autolag='AIC', store=False, regresults=False)

...statsmodels.tsa.stattools.adfuller(cc_cycle, maxlag=10, regression='c', autolag='AIC', store=False, regresults=False)

...statsmodels.tsa.stattools.adfuller(it_cycle, maxlag=10, regression='c', autolag='AIC', store=False, regresults=False)

...statsmodels.tsa.stattools.adfuller(xr_cycle, maxlag=10, regression='c', autolag='AIC', store=False, regresults=False)

...statsmodels.tsa.stattools.adfuller(ir_cycle, maxlag=10, regression='c', autolag='AIC', store=False, regresults=False)

...statsmodels.tsa.stattools.adfuller(inf_cycle, maxlag=10, regression='c', autolag='AIC', store=False, regresults=False)

...statsmodels.tsa.stattools.adfuller(con_cycle, maxlag=10, regression='c', autolag='AIC', store=False, regresults=False)

```
...statsmodels.tsa.stattools.adfuller(i_cycle, maxlag=10, regression='c', autolag='AIC', store=False, regresults=False)

...print('parece que todos los valores son estacionarios a diferente AIC, esto lleva un análisis más sensillo')

...get_ipython().magic('matplotlib inline')

import matplotlib.pyplot as plt

...fig2 = plt.figure(figsize=(40, 32))

plt.subplot(431)

plt.plot(y, 'b-', label='Serie original' )

plt.plot(y_trend, 'r-', label='Tendencia' )

plt.legend( loc='lower right' )

#plt.title(y.name,fontsize=25)

plt.title('PIB',fontsize=25)

plt.grid(True)

...plt.subplot(432)

plt.plot(g, 'b-', label='Serie original' )

plt.plot(g_trend, 'r-', label='Tendencia')

plt.legend( loc='lower right' )

plt.title('Gasto público neto pagado',fontsize=25)

#plt.title(g.name,fontsize=25)

plt.grid(True)

...plt.subplot(433)

plt.plot(gs, 'b-', label='Serie original')

plt.plot(gs_trend, 'r-', label='Tendencia')

plt.legend( loc='lower right' )
```

```
#plt.title(gs.name,fontsize=25)

plt.title('Gasto sin costo financiero ni adefas',fontsize=25)

plt.grid(True)

...plt.subplot(434)

plt.plot(gk, 'b-', label='Serie original')

plt.plot( gk_trend, 'r-', label='Tendencia')

plt.legend( loc='lower right' )

plt.title('Gasto de capital presupuestario',fontsize=25)

#plt.title(gk.name,fontsize=25)

plt.grid(True)

...plt.subplot(435)

plt.plot(gcs, 'b-', label='Serie original' )

plt.plot(gcs_trend, 'r-', label='Tendencia')

plt.legend( loc='lower right' )

plt.title('Gasto corriente sin costo financiero ni adefas',fontsize=25)

#plt.title(gcs.name,fontsize=25)

plt.grid(True)

...plt.subplot(436)

plt.plot(it, 'b-', label='Serie original' )

plt.plot(it_trend, 'r-', label='Tendencia')

plt.legend( loc='lower right' )

plt.title('Ingreso tributario más aportaciones',fontsize=25)

#plt.title(it.name,fontsize=25)
```

```
plt.grid(True)

...plt.subplot(437)

plt.plot(cc, 'b-', label='Serie original' )

plt.plot(cc_trend, 'r-', label='Tendencia')

plt.legend( loc='lower right' )

#plt.title('Balanza comercial real',fontsize=25)

plt.title('Balanza comercial',fontsize=25)

plt.grid(True)

...plt.subplot(438)

plt.plot(con, 'b-', label='Serie original' )

plt.plot(con_trend, 'r-', label='Tendencia' )

plt.legend( loc='lower right' )

#plt.title(con.name,fontsize=25)

plt.title('Consumo',fontsize=25)

plt.grid(True)

...plt.subplot(439)

plt.plot(i, 'b-', label='Serie original' )

plt.plot(i_trend, 'r-', label='Tendencia' )

plt.legend( loc='upper right' )

plt.title('Inversión',fontsize=25)

#plt.title(i.name,fontsize=25)

plt.grid(True)

...plt.subplot(4,3,10)

plt.plot(ir, 'b-', label='Serie original' )
```

```
plt.plot(ir_trend, 'r-', label='Tendencia' )

plt.legend( loc='lower right' )

#plt.title(ir.name,fontsize=25)

plt.title('Tasa de interés',fontsize=25)

plt.grid(True)

...plt.subplot(4,3,11)

plt.plot(xr, 'b-', label='Serie original' )

plt.plot(xr_trend, 'r-', label='Tendencia' )

plt.legend( loc='lower right' )

#plt.title(xr.name,fontsize=25)

plt.title('Tipo de cambio',fontsize=25)

plt.grid(True)

...plt.subplot(4,3,12)

plt.plot(inf_d, 'b-', label='Serie original' )

plt.plot(inf_trend, 'r-', label='Tendencia' )

plt.legend( loc='lower right' )

plt.title('Inflación',fontsize=25)

plt.grid(True)

...plt.show()

...fig2.savefig('filtrosHP.eps', format='eps', dpi=10000)

...get_ipython().magic('matplotlib inline')

import matplotlib.pyplot as plt

...fig3 = plt.figure(figsize=(40, 32))
```

```
...plt.subplot(431)

plt.plot(y_cycle, 'r-', label='')

plt.legend( loc='upper left' )

#plt.title(y_cycle.name,fontsize=25)

plt.title('PIB',fontsize=25)

plt.grid(True)

...plt.subplot(432)

plt.plot(g_cycle, 'r-', label='')

plt.legend( loc='upper left' )

#plt.title(g_cycle.name,fontsize=25)

plt.title('Gasto público neto pagado',fontsize=25)

plt.grid(True)

...plt.subplot(433)

plt.plot(gs_cycle, 'r-', label='')

plt.legend( loc='upper left' )

#plt.title(gs_cycle.name,fontsize=25)

plt.title('Gasto sin costo financiero ni adefas',fontsize=25)

plt.grid(True)

...plt.subplot(434)

plt.plot( gk_cycle, 'r-', label='')

plt.legend( loc='upper left' )

#plt.title(gk_cycle.name,fontsize=25)

plt.title('Gasto de capital presupuestario',fontsize=25)
```

```
plt.grid(True)

...plt.subplot(435)

plt.plot(gcs_cycle, 'r-', label='')

plt.legend( loc='upper left' )

#plt.title(gcs_cycle.name,fontsize=25)

plt.title('Gasto corriente sin costo financiero ni adefas',fontsize=25)

plt.grid(True)

...plt.subplot(436)

plt.plot(it_cycle, 'r-', label='')

plt.legend( loc='upper left' )

#plt.title(it_cycle.name,fontsize=25)

plt.title('Ingreso tributario más aportaciones',fontsize=25)

plt.grid(True)

...plt.subplot(437)

plt.plot(cc_cycle, 'r-', label='')

plt.legend( loc='upper left' )

#plt.title('Balanza comercial (ciclo)',fontsize=25)

plt.title('Balanza comercial',fontsize=25)

plt.grid(True)

...plt.subplot(438)

plt.plot(con_cycle, 'r-', label='')

plt.legend( loc='upper left' )

#plt.title(con_cycle.name,fontsize=25)

plt.title('Consumo',fontsize=25)
```

```
plt.grid(True)

...plt.subplot(439)

plt.plot(i_cycle, 'r-', label='')

plt.legend( loc='upper left' )

plt.title('Inversión',fontsize=25)

#plt.title(i_cycle.name,fontsize=25)

plt.grid(True)

...plt.subplot(4,3,10)

plt.plot(ir_cycle, 'r-', label='')

plt.legend( loc='upper left' )

#plt.title(ir_cycle.name,fontsize=25)

plt.title('Tasa de interés',fontsize=25)

plt.grid(True)

...plt.subplot(4,3,11)

plt.plot(xr_cycle, 'r-', label='')

plt.legend( loc='upper left' )

#plt.title(xr_cycle.name,fontsize=25)

plt.title('Tipo de cambio',fontsize=25)

plt.grid(True)

...plt.subplot(4,3,12)

plt.plot(inf_cycle, 'r-', label='')

plt.legend( loc='upper left' )

#plt.title('Inflación (ciclo)',fontsize=25)
```

```
plt.title('Inflación',fontsize=25)

plt.grid(True)

...plt.show()

...fig3.savefig('Seriesfinal.eps', format='eps', dpi=10000)

...exo = pandas.concat([PCF,C95,C08,EZP,VFQ,FCH], axis=1)

...# In[ ]:

...# In[ ]:

...# In[ ]:

...# In[ ]:

...# In[ ]:

...# In[ ]:

...def var_est(endo,exo,lags,promedio,shock_orthogonal): #Estima con "lags" rezagos un modelo var.   Soporta hasta 10 rezagos

...    #rezagos

    L1endo = endo.shift(+1)

    L2endo = endo.shift(+2)

    L3endo = endo.shift(+3)

    L4endo = endo.shift(+4)

    L5endo = endo.shift(+5)

    L6endo = endo.shift(+6)

    L7endo = endo.shift(+7)

    L8endo = endo.shift(+8)

    L9endo = endo.shift(+9)

    L10endo = endo.shift(+10)
```

```
...    if lags == 1:
        qdata_dependiente = pandas.concat([L1endo,exo], axis=1)
      else:
        if lags == 2:
          qdata_dependiente = pandas.concat([L1endo,L2endo,exo], axis=1)
        else:
          if lags == 3:
            qdata_dependiente = pandas.concat([L1endo,L2endo,L3endo,exo], axis=1)
          else:
            if lags == 4:
              qdata_dependiente = pandas.concat([L1endo,L2endo,L3endo,L4endo,exo], axis=1)
            else:
              if lags == 5:
                qdata_dependiente = pandas.concat([L1endo,L2endo,L3endo,L4endo,L5endo,exo], axis=1)
              else:
                if lags == 6:
                  qdata_dependiente = pandas.concat([L1endo,L2endo,L3endo,L4endo,L5endo,L6endo,exo], axis=1)
                else:
                  if lags == 7:
                    qdata_dependiente = pandas.concat([L1endo,L2endo,L3endo,L4endo,L5endo,L6endo,
                                                       L7endo,exo], axis=1)
                  else:
                    if lags == 8:
                      qdata_dependiente = pandas.concat([L1endo,L2endo,L3endo,L4endo,L5endo,L6endo,
```

```
                                        L7endo,L8endo,exo], axis=1)
                else:
                    if lags == 9:
                        qdata_dependiente = pandas.concat([L1endo,L2endo,L3endo,L4endo,L5endo,L6endo,
                                                           L7endo,L8endo,L9endo,exo], axis=1)
                    else:
                        qdata_dependiente = pandas.concat([L1endo,L2endo,L3endo,L4endo,L5endo,L6endo,
                                                           L7endo,L8endo,L9endo,L10endo,exo], axis=1)
...     [K,C] = qdata_dependiente.shape
    qdata_dependiente = numpy.array( qdata_dependiente[lags:K] )
    Endo = numpy.array( endo[lags:K] )
    [R,C] = qdata_dependiente.shape
    qdata_dependiente = numpy.c_[numpy.ones(R),qdata_dependiente]
    [R,C] = qdata_dependiente.shape
...     D = qdata_dependiente
    I = Endo
    beta = numpy.linalg.inv( D.T @ D ) @ D.T @ I
    I_hat = D @ beta
    e_hat = I - I_hat
    omega = numpy.cov(e_hat.T,bias=1)
    chol = numpy.linalg.cholesky(omega)
...     constantes = numpy.array([beta[0,:]]).  T
    [t,v_exo] = exo.shape
```

```
    coe_rezagos = beta[1:C-v_exo].  T

    [v_endo,t] = coe_rezagos.shape

    d={}

    for i in range(1,lags+1):

        d["phi{0}".format(i)] = coe_rezagos[:,v_endo*(i-1):(v_endo)*i]

...     #función de respuesta

    def impulso_respuesta(chol,d,promedio,shock_orthogonal,lags): #promedio es el vector promedio de las variables,

        #shock orthogonal es un vector 0 y 1 para el shock

        [R,C] = shock_orthogonal.shape

...         from collections import namedtuple

        Phi = namedtuple('Struct', d.keys())(*d.values())

...         if lags == 1:

            phi1 = Phi.phi1

            phi2 = numpy.zeros((R,R))

            phi3 = numpy.zeros((R,R))

            phi4 = numpy.zeros((R,R))

            phi5 = numpy.zeros((R,R))

            phi6 = numpy.zeros((R,R))

            phi7 = numpy.zeros((R,R))

            phi8 = numpy.zeros((R,R))

            phi9 = numpy.zeros((R,R))

            phi10 =numpy.zeros((R,R))

        else:
```

```
if lags == 2:
    phi1 = Phi.phi1
    phi2 = Phi.phi2
    phi3 = numpy.zeros((R,R))
    phi4 = numpy.zeros((R,R))
    phi5 = numpy.zeros((R,R))
    phi6 = numpy.zeros((R,R))
    phi7 = numpy.zeros((R,R))
    phi8 = numpy.zeros((R,R))
    phi9 = numpy.zeros((R,R))
    phi10 =numpy.zeros((R,R))
else:
    if lags == 3:
        phi1 = Phi.phi1
        phi2 = Phi.phi2
        phi3 = Phi.phi3
        phi4 = numpy.zeros((R,R))
        phi5 = numpy.zeros((R,R))
        phi6 = numpy.zeros((R,R))
        phi7 = numpy.zeros((R,R))
        phi8 = numpy.zeros((R,R))
        phi9 = numpy.zeros((R,R))
        phi10 =numpy.zeros((R,R))
    else:
```

```
if lags == 4:
    phi1 = Phi.phi1
    phi2 = Phi.phi2
    phi3 = Phi.phi3
    phi4 = Phi.phi4
    phi5 = numpy.zeros((R,R))
    phi6 = numpy.zeros((R,R))
    phi7 = numpy.zeros((R,R))
    phi8 = numpy.zeros((R,R))
    phi9 = numpy.zeros((R,R))
    phi10 =numpy.zeros((R,R))
else:
    if lags ==5:
        phi1 = Phi.phi1
        phi2 = Phi.phi2
        phi3 = Phi.phi3
        phi4 = Phi.phi4
        phi5 = Phi.phi5
        phi6 = numpy.zeros((R,R))
        phi7 = numpy.zeros((R,R))
        phi8 = numpy.zeros((R,R))
        phi9 = numpy.zeros((R,R))
        phi10 =numpy.zeros((R,R))
```

```
else:
    if lags ==6:
        phi1 = Phi.phi1
        phi2 = Phi.phi2
        phi3 = Phi.phi3
        phi4 = Phi.phi4
        phi5 = Phi.phi5
        phi6 = Phi.phi6
        phi7 = numpy.zeros((R,R))
        phi8 = numpy.zeros((R,R))
        phi9 = numpy.zeros((R,R))
        phi10 =numpy.zeros((R,R))
    else:
        if lags ==7:
            phi1 = Phi.phi1
            phi2 = Phi.phi2
            phi3 = Phi.phi3
            phi4 = Phi.phi4
            phi5 = Phi.phi5
            phi6 = Phi.phi6
            phi7 = Phi.phi7
            phi8 = numpy.zeros((R,R))
            phi9 = numpy.zeros((R,R))
```

```
        phi10 =numpy.zeros((R,R))
    else:
        if lags ==8:
            phi1 = Phi.phi1
            phi2 = Phi.phi2
            phi3 = Phi.phi3
            phi4 = Phi.phi4
            phi5 = Phi.phi5
            phi6 = Phi.phi6
            phi7 = Phi.phi7
            phi8 = Phi.phi8
            phi9 = numpy.zeros((R,R))
            phi10 =numpy.zeros((R,R))
        else:
            if lags ==9:
                phi1 = Phi.phi1
                phi2 = Phi.phi2
                phi3 = Phi.phi3
                phi4 = Phi.phi4
                phi5 = Phi.phi5
                phi6 = Phi.phi6
                phi7 = Phi.phi7
                phi8 = Phi.phi8
                phi9 = Phi.phi9
```

```
                phi10 =numpy.zeros((R,R))
            else:
                phi1 = Phi.phi1
                phi2 = Phi.phi2
                phi3 = Phi.phi3
                phi4 = Phi.phi4
                phi5 = Phi.phi5
                phi6 = Phi.phi6
                phi7 = Phi.phi7
                phi8 = Phi.phi8
                phi9 = Phi.phi9
                phi10 =Phi.phi10
...     z = numpy.zeros((21,R))
...     z0 = chol @ shock_orthogonal
        z[0,:]=z0.  T
...     z1 = phi1 @ z0
        z[1,:]=z1.  T
...     z2 = phi1 @ z1+phi2 @ z0
        z[2,:]=z2.  T
...     z3 = phi1 @ z2+phi2 @ z1+phi3 @ z0
        z[3,:]=z3.  T
...     z4 = phi1 @ z3+phi2 @ z2+phi3 @ z1+phi4 @ z0
        z[4,:]=z4.  T
```

```
...     z5 = phi1 @ z4+phi2 @ z3+phi3 @ z2+phi4 @ z1+phi5 @ z0

    z[5,:]=z5.  T

...     z6 = phi1 @ z5+phi2 @ z4+phi3 @ z3+phi4 @ z2+phi5 @ z1+phi6 @ z0

    z[6,:]=z6.  T

...     z7 = phi1 @ z6+phi2 @ z5+phi3 @ z4+phi4 @ z3+phi5 @ z2+phi6 @ z1 +phi7 @ z0

    z[7,:]=z7.  T

...     z8 = phi1 @ z7+phi2 @ z6+phi3 @ z5+phi4 @ z4+phi5 @ z3+phi6 @ z2 +phi7 @ z1 +phi8 @ z0

    z[8,:]=z8.  T

...     z9 = phi1 @ z8+phi2 @ z7+phi3 @ z6+phi4 @ z5+phi5 @ z4+phi6 @ z3 +phi7 @ z2 +phi8 @ z1 +phi9 @ z0

    z[9,:]=z9.  T

...     z10 = phi1 @ z9+phi2 @ z8+phi3 @ z7+phi4 @ z6+phi5 @ z5+phi6 @ z4 +phi7 @ z3 +phi8 @ z2 +phi9 @ z1 +phi10 @ z0

    z[10,:]=z10.  T

...     z11 = phi1 @ z10+phi2 @ z9+phi3 @ z8+phi4 @ z7+phi5 @ z6+phi6 @ z5 +phi7 @ z4 +phi8 @ z3 +phi9 @ z2 +phi10 @ z1

    z[11,:]=z11.  T

...     z12 = phi1 @ z11+phi2 @ z10+phi3 @ z9+phi4 @ z8+phi5 @ z7+phi6 @ z6 +phi7 @ z5 +phi8 @ z4 +phi9 @ z3 +phi10 @ z2

    z[12,:]=z12.  T

...     z13 = phi1 @ z12+phi2 @ z11+phi3 @ z10+phi4 @ z9+phi5 @ z8+phi6 @ z7 +phi7 @ z6 +phi8 @ z5 +phi9 @ z4 +phi10 @ z3

    z[13,:]=z13.  T

...     z14 = phi1 @ z13+phi2 @ z12+phi3 @ z11+phi4 @ z10+phi5 @ z9+phi6 @ z8 +phi7 @ z7 +phi8 @ z6 +phi9 @ z5 +phi10 @
z4

    z[14,:]=z14.  T

...     z15 = phi1 @ z14+phi2 @ z13+phi3 @ z12+phi4 @ z11+phi5 @ z10+phi6 @ z9 +phi7 @ z8 +phi8 @ z7 +phi9 @ z6 +phi10 @
z5

    z[15,:]=z15.  T
```

```
...     z16 = phi1 @ z15+phi2 @ z14+phi3 @ z13+phi4 @ z12+phi5 @ z11+phi6 @ z10 +phi7 @ z9 +phi8 @ z8 +phi9 @ z7 +phi10 @
z6

    z[16,:]=z16.  T

...     z17 = phi1 @ z16+phi2 @ z15+phi3 @ z14+phi4 @ z13+phi5 @ z12+phi6 @ z11 +phi7 @ z10 +phi8 @ z9 +phi9 @ z8 +phi10
@ z7

    z[17,:]=z17.  T

...     z18 = phi1 @ z17+phi2 @ z16+phi3 @ z15+phi4 @ z14+phi5 @ z13+phi6 @ z12 +phi7 @ z11 +phi8 @ z10 +phi9 @ z9 +phi10
@ z8

    z[18,:]=z18.  T

...     z19 = phi1 @ z18+phi2 @ z17+phi3 @ z16+phi4 @ z15+phi5 @ z14+phi6 @ z13 +phi7 @ z12 +phi8 @ z11 +phi9 @ z10
+phi10 @ z9

    z[19,:]=z19.  T

...     z20 = phi1 @ z19+phi2 @ z18+phi3 @ z17+phi4 @ z16+phi5 @ z15+phi6 @ z14 +phi7 @ z13 +phi8 @ z12 +phi9 @ z11
+phi10 @ z10

    z[20,:]=z20.  T

...     z_c = numpy.copy(z)

    z_mf = numpy.zeros((21,R))

...     for i in range(0,R):

        z_mf[:,i] = z_c[:,i]*promedio[i,0]

...     for j in range(0,R):

        for i in range(0,21):

            if i == 0:

                z_mf[i,j] = z_mf[i,j]

            else:

                z_mf[i,j] = z_mf[i,j] + z_mf[i-1,j]

...     MF = numpy.zeros((21, R))

    z_mf_shock_var = numpy.zeros((21, 1))
```

```
        z_mf_shock = numpy.zeros((21, R))
...         for i in range(0,R):
             z_mf_shock[:,i] = z_mf[:,i]*shock_orthogonal[i,0]
...         for i in range(0,21):
             z_mf_shock_var[i,0] = numpy.sum( z_mf_shock[i,:] )
...         for j in range(0,R):
             for i in range(0,21):
                MF[i,j] = z_mf[i,j]/z_mf_shock_var[i,0]
...         return MF,z
...     [MF_puntual,z_puntual] = impulso_respuesta(chol,d,promedio,shock_orthogonal,lags)
...     def boots(I,D,exo):
        [R_exo,C_exo] =exo.shape
        [R,C] = D.shape
        [N,K] = I.shape
        import random
        V = numpy.array([range(0,R)])
        Ran = numpy.random.randint(0,R,size=(R,1))
        I_aux = numpy.zeros(( R, K))
        D_aux = numpy.zeros(( R, C))
...         for i in V:
             I_aux[i,:] = I[Ran[i,0],:]
             D_aux[i,:] = D[Ran[i,0],:]
...         #que las exógenas no tengan problema de colinealidad
```

```
        flag = numpy.zeros(( 1,C_exo ))
        exo_aux = D_aux[:,C-C_exo:C]
...         for j in range(0,C_exo):
             flag[0,j] = numpy.linalg.norm( exo_aux[:,j] )
...         posi = numpy.array( numpy.where( flag>0) )
        [R_posi, C_posi] = posi.shape
        O = D_aux[:,0:C-C_exo]
        X_AUX = O
...         for j in range(0,C_posi):
             X_AUX = numpy.c_[ X_AUX,exo_aux[ :,posi[1,j] ] ]
...         X = X_AUX
...         return I_aux,X
...     chol_B = numpy.zeros(( v_endo, v_endo , bootstrap))
    MF_B = numpy.zeros(( 21, v_endo , bootstrap))
    z_B = numpy.zeros(( 21, v_endo , bootstrap))
...     for i in range(0,bootstrap):
        [IB,DB] = boots(I,D,exo)
        betaB = numpy.linalg.inv( DB.T @ DB ) @ DB.T @ IB
        IB_hat = DB @ betaB
        eB_hat = IB - IB_hat
        omegaB = numpy.cov(eB_hat.T,bias=1)
        cholB = numpy.linalg.cholesky(omegaB)
        chol_B[:,:,i]=cholB
        constantes = numpy.array([betaB[0,:]]).  T
```

```
        [t,v_exo] = exo.shape

        coe_rezagos = beta[1:C-v_exo].  T

        [v_endo,t] = coe_rezagos.shape

        d={}

...         for j in range(1,lags+1):

            d["phi{0}".format(j)] = coe_rezagos[:,v_endo*(j-1):(v_endo)*j]

...         [MF_b,z_b] = impulso_respuesta(cholB,d,promedio,shock_orthogonal,lags)

        MF_B[:,:,i] = MF_b

        z_B[:,:,i] = z_b

...     MF_alto = numpy.zeros(( 21, v_endo ))

    z_alto = numpy.zeros(( 21, v_endo ))

    MF_bajo = numpy.zeros(( 21, v_endo ))

    z_bajo = numpy.zeros(( 21, v_endo ))

...     for j in range(0,v_endo):

        for i in range(0,21):

            MF_alto[i,j] = numpy.percentile( MF_B[i,j,:], 97.5)

            MF_bajo[i,j] = numpy.percentile( MF_B[i,j,:], 2.5)

            z_alto[i,j] = numpy.percentile( z_B[i,j,:], 97.5)

            z_bajo[i,j] = numpy.percentile( z_B[i,j,:], 2.5)

...     return R,chol,d,MF_puntual,z_puntual,MF_alto,MF_bajo,z_alto,z_bajo,beta,d

...y_mean =numpy.mean(numpy.exp(y))

g_mean =numpy.mean(numpy.exp(g))

gc_mean =numpy.mean(numpy.exp(gc))
```

```
gk_mean =numpy.mean(numpy.exp(gk))

cc_mean =numpy.mean(cc)

it_mean =numpy.mean(numpy.exp(it))

gs_mean =numpy.mean(numpy.exp(gs))

gcs_mean =numpy.mean(numpy.exp(gcs))

con_mean =numpy.mean(numpy.exp(con))

ip_mean =numpy.mean(numpy.exp(ip))

i_mean =numpy.mean(numpy.exp(i))

ir_mean =numpy.mean(ir)

xr_mean =numpy.mean(xr)

inf_mean =numpy.mean(inf)

...#caso bivariado.   Shock del gasto público neto sobre el PIB (prueba de rezagos óptimos)

qdata = pandas.concat([gs_cycle,y_cycle], axis=1)

model = statsmodels.tsa.api.VAR(qdata)

results = model.fit(maxlags=10, ic='aic')

results.summary()

model.select_order(10)

...#caso bivariado.   Shock del gasto público neto sobre el PIB 2 rezagos óptimos para el ejercicio anterior

qdata = pandas.concat([gs_cycle,y_cycle], axis=1)

exo = pandas.concat([PCF,C95,C08,EZP,VFQ,FCH], axis=1)

#exo = pandas.concat([C95,C08], axis=1)

promedio = numpy.array([ [gs_mean], [y_mean] ])

shock_orthogonal = numpy.array([ [1], [0] ])
```

```
...[R,chol,d,MF_puntual,z_puntual,MF_alto,MF_bajo,z_alto,z_bajo,beta,d] = var_est(qdata,exo,2,promedio,shock_orthogonal)

...get_ipython().magic('matplotlib inline')

import matplotlib.pyplot as plt

fig = plt.figure(figsize=(10, 10))

x0 = fig.add_subplot(111)

xf = fig.add_subplot(111)

...plt.plot(MF_bajo[:,1], 'b--',label='Intervalo de confianza al 95%' )

plt.plot(MF_alto[:,1], 'b--')

plt.plot(MF_puntual[:,1], 'r-',label='Estimación puntual' )

plt.legend( loc='lower left' )

plt.grid(True)

...x0.annotate( round(MF_puntual[0,1], 2), xy=(2, 1), xytext=(0, 0),fontsize=15)

xf.annotate( round(MF_puntual[20,1], 2), xy=(2, 1), xytext=(18, -1),fontsize=15)

...plt.show()

fig.savefig('Impulso_respuesta_general.eps', format='eps', dpi=10000)

MF_puntual

...#caso bivariado.   Shock del gasto público neto sobre el PIB 2 rezagos óptimos para el ejercicio anterior

qdata = pandas.concat([gs_cycle,y_cycle], axis=1)

exo = pandas.concat([PCF,C95,C08,EZP,VFQ,FCH], axis=1)

promedio = numpy.array([ [gs_mean], [y_mean] ])

shock_orthogonal = numpy.array([ [1], [0] ])

...[R,chol,d,MF_puntual,z_puntual,MF_alto,MF_bajo,z_alto,z_bajo,beta,d] = var_est(qdata,exo,4,promedio,shock_orthogonal)

[R2,chol2,d2,MF_puntual2,z_puntual2,MF_alto2,MF_bajo2,z_alto2,z_bajo2,beta2,d2] =
var_est(qdata,exo,6,promedio,shock_orthogonal)
```

```
...get_ipython().magic('matplotlib inline')

import matplotlib.pyplot as plt

fig = plt.figure(figsize=(15, 5))

x01 = fig.add_subplot(121)

x02 = fig.add_subplot(122)

xf1 = fig.add_subplot(121)

xf2 = fig.add_subplot(122)

...plt.subplot(121)

x01.annotate( round(MF_puntual[0,1], 2), xy = (.1,.1),xytext=(0, 0),fontsize=15)

xf1.annotate( round(MF_puntual[20,1], 2), xy = (.1,.1),xytext=(18, -1.5),fontsize=15)

plt.plot(MF_bajo[:,1], 'b--',label='Intervalo de confianza al 95%')

plt.plot(MF_alto[:,1], 'b--')

plt.plot(MF_puntual[:,1], 'r-',label='Estimación puntual' )

plt.legend( loc='lower left' )

plt.title('4 rezagos')

plt.grid(True)

...plt.subplot(122)

x02.annotate( round(MF_puntual2[0,1], 2), xy = (.1,.1),xytext=(0, 0),fontsize=15)

xf2.annotate( round(MF_puntual2[20,1], 2), xy = (.1,.1),xytext=(18, -2),fontsize=15)

plt.plot(MF_bajo2[:,1], 'b--',label='Intervalo de confianza al 95%')

plt.plot(MF_alto2[:,1], 'b--')

plt.plot(MF_puntual2[:,1], 'r-',label='Estimación puntual' )

plt.legend( loc='lower left' )

plt.title('6 rezagos')
```

```
plt.grid(True)

...plt.show()

fig.savefig('Impulso_respuesta_general2.eps', format='eps', dpi=10000)

...export = qdata = pandas.concat([gs_cycle,y_cycle,exo], axis=1)

...chol

...#caso multivariado.   Shock del gasto público neto sobre el PIB

qdata = pandas.concat([gs_cycle,con_cycle], axis=1)

exo = pandas.concat([ PCF,C95,C08,EZP,VFQ,FCH ], axis=1)

promedio = numpy.array([ [gs_mean], [con_mean] ])

shock_orthogonal = numpy.array([ [1], [0] ])

[R,chol,d,MF_puntual,z_puntual,MF_alto,MF_bajo,z_alto,z_bajo,beta,d] = var_est(qdata,exo,2,promedio,shock_orthogonal)

...qdata = pandas.concat([gs_cycle,i_cycle], axis=1)

exo = pandas.concat([ PCF,C95,C08,EZP,VFQ,FCH ], axis=1)

promedio = numpy.array([ [gs_mean], [i_mean] ])

shock_orthogonal = numpy.array([ [1], [0] ])

[R2,chol2,d2,MF_puntual2,z_puntual2,MF_alto2,MF_bajo2,z_alto2,z_bajo2,beta2,d2] =
var_est(qdata,exo,2,promedio,shock_orthogonal)

...get_ipython().magic('matplotlib inline')

import matplotlib.pyplot as plt

fig = plt.figure(figsize=(15, 5))

x01 = fig.add_subplot(121)

xf1 = fig.add_subplot(121)

x02 = fig.add_subplot(122)

xf2 = fig.add_subplot(122)
```

```
...plt.subplot(121)

x01.annotate( round(z_puntual[0,1], 2), xy = (.1,.1),xytext=(0, 0),fontsize=15)

xf1.annotate( round(z_puntual[20,1], 2), xy = (.1,.1),xytext=(18, -.75),fontsize=15)

plt.plot(z_bajo[:,1], 'b--',label='Intervalo de confianza al 95%')

plt.plot(z_alto[:,1], 'b--')

plt.plot(z_puntual[:,1], 'r-',label='Estimación puntual' )

plt.legend( loc='lower right' )

plt.title('Consumo (2 rezagos)')

plt.grid(True)

...plt.subplot(122)

x02.annotate( round(z_puntual2[0,1], 2), xy = (.1,.1),xytext=(0, 0),fontsize=15)

xf2.annotate( round(z_puntual2[20,1], 2), xy = (.1,.1),xytext=(18, .5),fontsize=15)

plt.plot(z_bajo2[:,1], 'b--',label='Intervalo de confianza al 95%')

plt.plot(z_alto2[:,1], 'b--')

plt.plot(z_puntual2[:,1], 'r-',label='Estimación puntual' )

plt.legend( loc='lower right' )

plt.title('Inversión privada (2 rezagos)')

plt.grid(True)

...plt.show()

fig.savefig('MFbivar2.eps', format='eps', dpi=10000)

MF_alto

...# In[ ]:

...# In[ ]:
```

```
...qdata = pandas.concat([gk_cycle,gcs_cycle,y_cycle,cc_cycle], axis=1)

model = statsmodels.tsa.api.VAR(qdata)

results = model.fit(maxlags=10, ic='aic')

results.summary()

model.select_order(10)

...#caso multivariado.   Shock del gasto público neto sobre el PIB

qdata = pandas.concat([gk_cycle,gcs_cycle,y_cycle,cc_cycle], axis=1)

exo = pandas.concat([ PCF,C95,C08,EZP,VFQ,FCH ], axis=1)

promedio = numpy.array([ [gk_mean], [gcs_mean], [y_mean], [cc_mean]  ])

shock_orthogonal = numpy.array([ [1], [0], [0], [0] ])

[R,chol,d,MF_puntual,z_puntual,MF_alto,MF_bajo,z_alto,z_bajo,beta,d] = var_est(qdata,exo,2,promedio,shock_orthogonal)

shock_orthogonal = numpy.array([ [0], [1], [0], [0] ])

[R2,chol2,d2,MF_puntual2,z_puntual2,MF_alto2,MF_bajo2,z_alto2,z_bajo2,beta2,d2] =
var_est(qdata,exo,2,promedio,shock_orthogonal)

...get_ipython().magic('matplotlib inline')

import matplotlib.pyplot as plt

fig = plt.figure(figsize=(15, 5))

x01 = fig.add_subplot(121)

x02 = fig.add_subplot(122)

xf1 = fig.add_subplot(121)

xf2 = fig.add_subplot(122)

...plt.subplot(121)

x01.annotate( round(MF_puntual[0,2], 2), xy = (.1,.1),xytext=(0, 0),fontsize=15)

xf1.annotate( round(MF_puntual[20,2], 2), xy = (.1,.1),xytext=(18, -1.5),fontsize=15)
```

```
plt.plot(MF_bajo[:,2], 'b--',label='Intervalo de confianza al 95%')

plt.plot(MF_alto[:,2], 'b--')

plt.plot(MF_puntual[:,2], 'r-',label='Estimación puntual' )

plt.legend( loc='lower left' )

plt.title('Choque al gasto en capital (2 rezagos)')

plt.grid(True)

...plt.subplot(122)

x02.annotate( round(MF_puntual2[0,2], 2), xy = (.1,.1),xytext=(0, 0),fontsize=15)

xf2.annotate( round(MF_puntual2[20,2], 2), xy = (.1,.1),xytext=(18, 0.5),fontsize=15)

plt.plot(MF_bajo2[:,2], 'b--',label='Intervalo de confianza al 95%')

plt.plot(MF_alto2[:,2], 'b--')

plt.plot(MF_puntual2[:,2], 'r-',label='Estimación puntual' )

plt.legend( loc='lower left' )

plt.title('Choque al gasto corriente (2 rezagos)')

plt.grid(True)

...plt.show()

fig.savefig('MF4var.eps', format='eps', dpi=10000)

...#caso multivariado.   Shock del gasto público neto sobre el PIB

qdata = pandas.concat([gk_cycle,gcs_cycle,y_cycle,cc_cycle], axis=1)

exo = pandas.concat([ PCF,C95,C08,EZP,VFQ,FCH ], axis=1)

promedio = numpy.array([ [gk_mean], [gcs_mean], [y_mean], [cc_mean]  ])

shock_orthogonal = numpy.array([ [1], [0], [0], [0] ])

[R,chol,d,MF_puntual,z_puntual,MF_alto,MF_bajo,z_alto,z_bajo,beta,d] = var_est(qdata,exo,4,promedio,shock_orthogonal)
```

```
shock_orthogonal = numpy.array([ [0], [1], [0], [0] ])

[R2,chol2,d2,MF_puntual2,z_puntual2,MF_alto2,MF_bajo2,z_alto2,z_bajo2,beta2,d2] =
var_est(qdata,exo,4,promedio,shock_orthogonal)

...get_ipython().magic('matplotlib inline')

import matplotlib.pyplot as plt

fig = plt.figure(figsize=(15, 5))

x01 = fig.add_subplot(121)

x02 = fig.add_subplot(122)

xf1 = fig.add_subplot(121)

xf2 = fig.add_subplot(122)

...plt.subplot(121)

x01.annotate( round(MF_puntual[0,2], 2), xy = (.1,.1),xytext=(0, 0),fontsize=15)

xf1.annotate( round(MF_puntual[20,2], 2), xy = (.1,.1),xytext=(18, -1.5),fontsize=15)

plt.plot(MF_bajo[:,2], 'b--',label='Intervalo de confianza al 95%')

plt.plot(MF_alto[:,2], 'b--')

plt.plot(MF_puntual[:,2], 'r-',label='Estimación puntual' )

plt.legend( loc='lower left' )

plt.title('Choque al gasto en capital (4 rezagos)')

plt.grid(True)

...plt.subplot(122)

x02.annotate( round(MF_puntual2[0,2], 2), xy = (.1,.1),xytext=(0, 0),fontsize=15)

xf2.annotate( round(MF_puntual2[20,2], 2), xy = (.1,.1),xytext=(18, -2),fontsize=15)

plt.plot(MF_bajo2[:,2], 'b--',label='Intervalo de confianza al 95%')

plt.plot(MF_alto2[:,2], 'b--')
```

```
plt.plot(MF_puntual2[:,2], 'r-',label='Estimación puntual' )

plt.legend( loc='lower left' )

plt.title('Choque al gasto corriente (4 rezagos)')

plt.grid(True)

...plt.show()

fig.savefig('MF4var2.eps', format='eps', dpi=10000)

...# In[ ]:

...# In[ ]:

...qdata = pandas.concat([gk_cycle,gcs_cycle,y_cycle,xr_cycle,ir_cycle,cc_cycle,con_cycle,inf_cycle], axis=1)

model = statsmodels.tsa.api.VAR(qdata)

results = model.fit(maxlags=5, ic='aic')

results.summary()

model.select_order(5)

...#caso multivariado.   Shock del gasto público neto sobre el PIB

qdata = pandas.concat([gk_cycle,gcs_cycle,y_cycle,xr_cycle,ir_cycle,cc_cycle,inf_cycle], axis=1)

exo = pandas.concat([ PCF,C95,C08,EZP,VFQ,FCH ], axis=1)

promedio = numpy.array([ [gk_mean], [gcs_mean], [y_mean], [xr_mean], [ir_mean], [cc_mean], [con_mean], [inf_mean]  ])

shock_orthogonal = numpy.array([ [1], [0], [0], [0], [0], [0], [0] ])

[R,chol,d,MF_puntual,z_puntual,MF_alto,MF_bajo,z_alto,z_bajo,beta,d] = var_est(qdata,exo,2,promedio,shock_orthogonal)

shock_orthogonal = numpy.array([ [0], [1], [0], [0], [0],[0], [0] ])

[R2,chol2,d2,MF_puntual2,z_puntual2,MF_alto2,MF_bajo2,z_alto2,z_bajo2,beta2,d2] =
var_est(qdata,exo,2,promedio,shock_orthogonal)

...get_ipython().magic('matplotlib inline')

import matplotlib.pyplot as plt
```

```
fig = plt.figure(figsize=(15, 5))

x01 = fig.add_subplot(121)

x02 = fig.add_subplot(122)

xf1 = fig.add_subplot(121)

xf2 = fig.add_subplot(122)

...plt.subplot(121)

x01.annotate( round(MF_puntual[0,2], 2), xy = (.1,.1),xytext=(0, 0),fontsize=15)

xf1.annotate( round(MF_puntual[20,2], 2), xy = (.1,.1),xytext=(18, 0),fontsize=15)

plt.plot(MF_bajo[:,2], 'b--',label='Intervalo de confianza al 95%')

plt.plot(MF_alto[:,2], 'b--')

plt.plot(MF_puntual[:,2], 'r-',label='Estimación puntual' )

plt.legend( loc='lower left' )

plt.title('Choque al gasto en capital (2 rezagos)')

plt.grid(True)

...plt.subplot(122)

x02.annotate( round(MF_puntual2[0,2], 2), xy = (.1,.1),xytext=(0, 0),fontsize=15)

xf2.annotate( round(MF_puntual2[20,2], 2), xy = (.1,.1),xytext=(18, 0),fontsize=15)

plt.plot(MF_bajo2[:,2], 'b--',label='Intervalo de confianza al 95%')

plt.plot(MF_alto2[:,2], 'b--')

plt.plot(MF_puntual2[:,2], 'r-',label='Estimación puntual' )

plt.legend( loc='lower left' )

plt.title('Choque al gasto corriente (2 rezagos)')

plt.grid(True)

...plt.show()
```

```
fig.savefig('MF4var3.eps', format='eps', dpi=10000)

...#caso multivariado.   Shock del gasto público neto sobre el PIB

qdata = pandas.concat([gk_cycle,gcs_cycle,con_cycle,xr_cycle,ir_cycle,cc_cycle, inf_cycle, y_cycle], axis=1)

exo = pandas.concat([ PCF,C95,C08,EZP,VFQ,FCH ], axis=1)

promedio = numpy.array([ [gk_mean], [gcs_mean], [con_mean], [xr_mean], [ir_mean], [cc_mean], [inf_mean], [y_mean]  ])

shock_orthogonal = numpy.array([ [1], [0], [0], [0], [0], [0], [0], [0] ])

[R,chol,d,MF_puntual,z_puntual,MF_alto,MF_bajo,z_alto,z_bajo,beta,d] = var_est(qdata,exo,2,promedio,shock_orthogonal)

shock_orthogonal = numpy.array([ [0], [1], [0], [0], [0], [0], [0], [0] ])

[R2,chol2,d2,MF_puntual2,z_puntual2,MF_alto2,MF_bajo2,z_alto2,z_bajo2,beta2,d2] =
var_est(qdata,exo,2,promedio,shock_orthogonal)

...get_ipython().magic('matplotlib inline')

import matplotlib.pyplot as plt

fig = plt.figure(figsize=(15, 5))

x01 = fig.add_subplot(121)

x02 = fig.add_subplot(122)

xf1 = fig.add_subplot(121)

xf2 = fig.add_subplot(122)

...plt.subplot(121)

x01.annotate( round(z_puntual[0,2], 2), xy = (.1,.1),xytext=(0, 0),fontsize=15)

xf1.annotate( round(z_puntual[20,2], 2), xy = (.1,.1),xytext=(18, 0),fontsize=15)

plt.plot(z_bajo[:,2], 'b--',label='Intervalo de confianza al 95%')

plt.plot(z_alto[:,2], 'b--')

plt.plot(z_puntual[:,2], 'r-',label='Estimación puntual' )

plt.legend( loc='lower right' )
```

```
plt.title('Choque al gasto en capital (2 rezagos)')

plt.grid(True)

...plt.subplot(122)

x02.annotate( round(z_puntual2[0,2], 2), xy = (.1,.1),xytext=(0, 0),fontsize=15)

xf2.annotate( round(z_puntual2[20,2], 2), xy = (.1,.1),xytext=(18, 0),fontsize=15)

plt.plot(z_bajo2[:,2], 'b--',label='Intervalo de confianza al 95%')

plt.plot(z_alto2[:,2], 'b--')

plt.plot(z_puntual2[:,2], 'r-',label='Estimación puntual' )

plt.legend( loc='lower right' )

plt.title('Choque al gasto corriente (2 rezagos)')

plt.grid(True)

...plt.show()

fig.savefig('MF4varCon.eps', format='eps', dpi=10000)

...#caso multivariado.   Shock del gasto público neto sobre el PIB

qdata = pandas.concat([gk_cycle,gcs_cycle,i_cycle,xr_cycle,ir_cycle,cc_cycle, inf_cycle, y_cycle], axis=1)

exo = pandas.concat([ PCF,C95,C08,EZP,VFQ,FCH ], axis=1)

promedio = numpy.array([ [gk_mean], [gcs_mean], [i_mean], [xr_mean], [ir_mean], [cc_mean], [inf_mean], [y_mean] ])

shock_orthogonal = numpy.array([ [1], [0], [0], [0], [0], [0], [0], [0] ])

[R,chol,d,MF_puntual,z_puntual,MF_alto,MF_bajo,z_alto,z_bajo,beta,d] = var_est(qdata,exo,2,promedio,shock_orthogonal)

shock_orthogonal = numpy.array([ [0], [1], [0], [0], [0], [0], [0], [0] ])

[R2,chol2,d2,MF_puntual2,z_puntual2,MF_alto2,MF_bajo2,z_alto2,z_bajo2,beta2,d2] =
var_est(qdata,exo,2,promedio,shock_orthogonal)

...get_ipython().magic('matplotlib inline')

import matplotlib.pyplot as plt
```

```
fig = plt.figure(figsize=(15, 5))

...plt.subplot(121)

plt.plot(z_bajo[:,2], 'b--',label='Intervalo de confianza al 95%')

plt.plot(z_alto[:,2], 'b--')

plt.plot(z_puntual[:,2], 'r-',label='Estimación puntual' )

plt.legend( loc='lower right' )

plt.title('Choque al gasto en capital (2 rezagos)')

plt.grid(True)

...plt.subplot(122)

plt.plot(z_bajo2[:,2], 'b--',label='Intervalo de confianza al 95%')

plt.plot(z_alto2[:,2], 'b--')

plt.plot(z_puntual2[:,2], 'r-',label='Estimación puntual' )

plt.legend( loc='lower right' )

plt.title('Choque al gasto corriente (2 rezagos)')

plt.grid(True)

...plt.show()

fig.savefig('MF4varI.eps', format='eps', dpi=10000)

...qdata = pandas.concat([gs_cycle,it_cycle,y_cycle], axis=1)

model = statsmodels.tsa.api.VAR(qdata)

results = model.fit(maxlags=10, ic='aic')

results.summary()

model.select_order(10)

...# In[ ]:
```

```
...#caso multivariado.   Shock del gasto público neto sobre el PIB

qdata = pandas.concat([gs_cycle,it_cycle,y_cycle], axis=1)

exo = pandas.concat([ PCF,C95,C08,EZP,VFQ,FCH ], axis=1)

promedio = numpy.array([ [gs_mean], [it_mean], [y_mean] ])

shock_orthogonal = numpy.array([ [1], [0], [0] ])

[R,chol,d,MF_puntual,z_puntual,MF_alto,MF_bajo,z_alto,z_bajo,beta,d] = var_est(qdata,exo,2,promedio,shock_orthogonal)

shock_orthogonal = numpy.array([ [0], [1], [0] ])

[R2,chol2,d2,MF_puntual2,z_puntual2,MF_alto2,MF_bajo2,z_alto2,z_bajo2,beta2,d2] =
var_est(qdata,exo,2,promedio,shock_orthogonal)

shock_orthogonal = numpy.array([ [1], [0], [0] ])

[R3,chol3,d3,MF_puntual3,z_puntual3,MF_alto3,MF_bajo3,z_alto3,z_bajo3,beta3,d3] =
var_est(qdata,exo,6,promedio,shock_orthogonal)

shock_orthogonal = numpy.array([ [0], [1], [0] ])

[R4,chol4,d4,MF_puntual4,z_puntual4,MF_alto4,MF_bajo4,z_alto4,z_bajo4,beta4,d4] =
var_est(qdata,exo,6,promedio,shock_orthogonal)

...get_ipython().magic('matplotlib inline')

import matplotlib.pyplot as plt

fig = plt.figure(figsize=(15, 10))

x01 = fig.add_subplot(221)

x02 = fig.add_subplot(222)

xf1 = fig.add_subplot(221)

xf2 = fig.add_subplot(222)

...x03 = fig.add_subplot(223)

x04 = fig.add_subplot(224)

xf3 = fig.add_subplot(223)
```

```
xf4 = fig.add_subplot(224)

...plt.subplot(221)

x01.annotate( round(MF_puntual[0,2], 2), xy = (.1,.1),xytext=(0, 0),fontsize=15)

xf1.annotate( round(MF_puntual[20,2], 2), xy = (.1,.1),xytext=(18, -.5),fontsize=15)

plt.plot(MF_bajo[:,2], 'b--',label='Intervalo de confianza al 95%')

plt.plot(MF_alto[:,2], 'b--')

plt.plot(MF_puntual[:,2], 'r-',label='Estimación puntual' )

plt.legend( loc='lower left' )

plt.title('Choque positivo al gasto público (2 rezagos)')

plt.grid(True)

...plt.subplot(222)

x02.annotate( round(MF_puntual2[0,2], 2), xy = (.1,.1),xytext=(0, 0),fontsize=15)

xf2.annotate( round(MF_puntual2[20,2], 2), xy = (.1,.1),xytext=(18, -.2),fontsize=15)

plt.plot(MF_bajo2[:,2], 'b--',label='Intervalo de confianza al 95%')

plt.plot(MF_alto2[:,2], 'b--')

plt.plot(MF_puntual2[:,2], 'r-',label='Estimación puntual' )

plt.legend( loc='lower left' )

plt.title('Choque positivo a los ingresos tributarios (2 rezagos)')

plt.grid(True)

...plt.subplot(223)

x03.annotate( round(MF_puntual3[0,2], 2), xy = (.1,.1),xytext=(0, 0),fontsize=15)

xf3.annotate( round(MF_puntual3[20,2], 2), xy = (.1,.1),xytext=(18, -2.5),fontsize=15)

plt.plot(MF_bajo3[:,2], 'b--',label='Intervalo de confianza al 95%')

plt.plot(MF_alto3[:,2], 'b--')
```

```
plt.plot(MF_puntual3[:,2], 'r-',label='Estimación puntual' )

plt.legend( loc='lower left' )

plt.title('Choque positivo al gasto público (6 rezagos)')

plt.grid(True)

...plt.subplot(224)

x04.annotate( round(MF_puntual4[0,2], 2), xy = (.1,.1),xytext=(0, 0),fontsize=15)

xf4.annotate( round(MF_puntual4[20,2], 2), xy = (.1,.1),xytext=(18, -1.5),fontsize=15)

plt.plot(MF_bajo4[:,2], 'b--',label='Intervalo de confianza al 95%')

plt.plot(MF_alto4[:,2], 'b--')

plt.plot(MF_puntual4[:,2], 'r-',label='Estimación puntual' )

plt.legend( loc='lower left' )

plt.title('Choque positivo a los ingresos tributarios (6 rezagos)')

plt.grid(True)

...plt.show()

fig.savefig('MF4var4.eps', format='eps', dpi=10000)
```