CapitalistConsumption

December 3, 2019

1 TODO

- [ ] Plug inflation in housing stock
- [ ] Add other autonomous expenditure

[1]:
```
#!pip install pysolve3
```

Time checking:

[2]:
```
from datetime import datetime
t1 = datetime.now()
```

2 Introduction

Jupyter notebook for the simplest model scratch. By convention, all exogenous variables are presented with a overline line (*e.g.* $\overline{a}$) and the parameters are represented by greek letters (*e.g.* α). The code in this document is executable and is strongly recommend to do the follow to ensure that the output is corrected and updated:

Run time > Restart and run all…

3 Changes

In this version, the households are splitted in workers and capitalists where workers spend all their income white capitalists spend a share o distributed profits.

4 Equations

4.1 General Equations

$$Y = C + I$$

$$C = C_w + C_k$$

$$I = I_f + I_h$$

$$\omega = \overline{\omega}$$

$$W = \omega \cdot Y$$

$$Y_K = \frac{K_f}{\overline{v}}$$

$$u = \frac{Y}{Y_K}$$

$$g_k = \frac{h \cdot u}{v}$$

$$Z = I_h$$

$$K = K_f + K_{HD}$$

4.2 Households

$$C_w = \alpha \cdot W$$

$$C_k = \alpha_2 \cdot FD$$

$$YD_w = W$$

$$YD_k = FD + \overline{r}_{m-1} \cdot M_{-1} - \overline{r}_{mo-1} \cdot MO$$

$$Sh_w = YD_w - C_w = 0$$

$$Sh_k = YD_k - C_k = \Delta M$$

$$\Delta MO = I_h$$

$$V_h = M + K_H - MO$$

$$NFW_h = S_h - I_h$$

$$\text{Residual} = \Delta M - \Delta L_f$$

4.3 Firms

$$\Delta L_f = I_f - FU$$

$$FT = Y - W = FU + FD$$

$$FU = \gamma_F \cdot (FT - r_{L_{-1}} \cdot L_{f_{-1}})$$

$$FD = (1 - \gamma_F) \cdot (FT - r_{L_{-1}} \cdot L_{f_{-1}})$$

$$I_f = h \cdot Y$$

$$\Delta K_f = I_f$$

$$\Delta h = h_{-1} \cdot \gamma_u \cdot (u - \overline{u}_n)$$

$$V_f = K_f - L_f$$

$$NFW_f = FU - I_f$$

4.4 Banks

$$L = L_f$$

$$\Delta M = \Delta L + \Delta MO$$

$$NFW_b = rl_{-1} \cdot L_{-1} + rmo_{-1} \cdot MO_{-1} - rm_{-1} \cdot M_{-1}$$

$$r_l = r_m + spread_l (= 0)$$

$$r_{mo} = r_m + spread_{mo} (= 0)$$

$$V_b = L + MO - M$$

4.5 Residential Investment

$$K_{HS} = K_{HD}$$

$$I_h^S = I_h$$

$$\Delta K_{HD} = I_h$$

$$I_h = (1 + g_Z) \cdot I_{h_{-1}}$$

$$k = \frac{K_h \cdot p_h}{K}$$

$$g_Z = \phi_0 - \phi_1 \cdot own$$

$$own = \left(\frac{1 + r_{mo}}{1 + \dot{p}}\right) - 1$$

5 Simulation setup

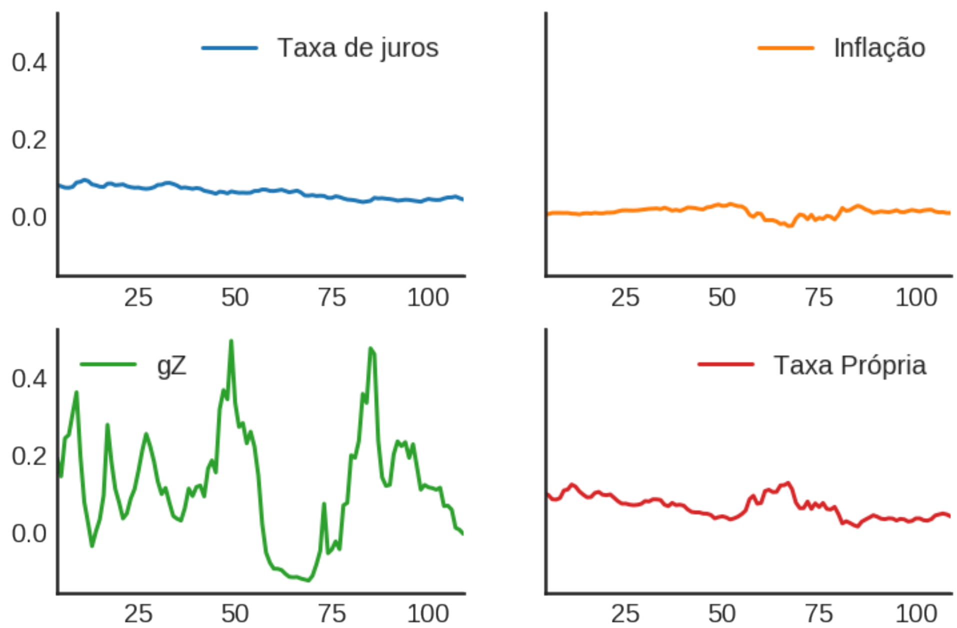
5.1 Loading libraries

[3]:
```
%config InlineBackend.figure_format = 'retina'

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patheffects as pe
import seaborn as sns
import networkx as nx
import sympy as sp
from sympy import pprint, cse

import ipywidgets as widgets
from ipywidgets import interact, interactive, fixed, interact_manual


plt.style.use('seaborn-white')

from pysolve3.model import Model
from pysolve3.utils import SolveSFC, ShockModel, SummaryShock, SFCTable
```

[4]:
```
data = pd.read_csv('../SeriesTemporais/Dados_yeojohnson.csv')

data = data[[
    "Taxa de juros",
    "Inflação",
```

```
    "gZ",
    "Crise",
    "Taxa Própria",
]]

data = data.dropna()

fig, ax = plt.subplots()

data.drop('Crise', axis='columns').plot(ax=ax, subplots=True, layout=(2,2),␣
 ↪sharex=False, sharey=True)
ax.axhline(y=0, ls='--', color='black')
sns.despine()
plt.show()
fig.savefig('YeoJohnson.png', dpi=300, bbox_inches = 'tight',)
```

```
/home/gpetrini/.local/lib/python3.6/site-
packages/pandas/plotting/_matplotlib/__init__.py:62: UserWarning: To output
multiple subplots, the figure containing the passed axes is being cleared
  plot_obj.generate()
```



6 Creating model function

[7]:
```
def model(
    alpha = 1,
    alpha_2 = .3,
    gamma_F = 0.9,
    gamma_u = 0.01,
    g_Z = 0.05,
    omega = 0.3,
    rm = 0.02,
    spread_l = 0,
    spread_mo = 0,
    un = 0.8,
    v = 2.5,
    phi_0 = 0.04,
    phi_1 = 0.02,
    infla = 0.0,
    phparam=1.0
):
  """
  phparam: 1.0 means no inflation
  """
  model = Model()
  model.set_var_default(0)
  model.var('C', desc='Total Consumption')
  model.var('C_w', desc='Workers Consumption')
  model.var('C_k', desc='Capitalists Consumption')
  model.var('FD', desc='Distributed profits')
  model.var('Fn', desc='Net profits')
  model.var('FT', desc='Total Profits')
  model.var('FU', desc='Retained profits')
  model.var('gk', desc='Capital growth rate', default=0.01)
  model.var('g_Z', desc='Autonomous grouth rate', default=g_Z)
  model.var('h', desc='Marginal propensity to invest (non-residential)',␣
↪default=0.30)
  model.var('I_t', desc='Investment', default = 100) # 200
  model.var('I_f', desc='Non-residential investment') # 100
  model.var('I_h', desc='Residential investment', default = 100) # 100
  model.var('Is', desc='Residential investment (Supply)', default = 100) # 100
  model.var('K_HS', desc='Houses supply', default=500) # 500
  model.var('K_HD', desc='Houses demand', default=500) # 500
  model.var('K_f', desc='Non-residential capital', default = 1000) # 10000
  model.var('Knom', desc='Nominal Capital', default=1500)
  model.var('K', desc='Real Capital', default=1500)
  model.var('K_k', desc="% of Kf in total")
  model.var('L', desc='Total Loans') # 100
  model.var('Lf', desc='Firms Loans') # 100
```

```
model.var('M', desc='Money deposits', default = 310) # 300
model.var('MO', desc='Mortgages', default = 300) # 200
model.var('NFW_h', desc='Households Net Financial Wealth')
model.var('NFW_f', desc='Firms Net Financial Wealth')
model.var('NFW_b', desc='Banks Net Financial Wealth')
model.var('own', desc='Own interest rate', default = 0.02)
model.var('ph', desc='House price', default = 1)
model.var('rl', desc='Interests rates on loans')
model.var('rmo', desc='Interests rates on mortgages')
model.var('Sh_w', desc='Workers savings', default = 0)
model.var('Sh_k', desc='Capitalist savings', default = 100)
model.var('u', desc='Capacity utilization ratio', default=0.7)
model.var('V_h', desc='Household net financial wealth', default = 500)
model.var('V_f', desc='Firms net financial wealth', default = 1000)
model.var('V_b', desc='Banks net financial wealth')
model.var('W', desc='Wages')
model.var('Y', desc='GDP')
model.var('Yk', desc='Capacity')
model.var('YD_w', desc='Workers disposable income')
model.var('YD_k', desc='Capitalists disposable income')
model.var('Z', desc='Autonomous expenditures')
model.var('Residual', desc='Unecessarily equation. Should be zero')

model.param('alpha', desc='Propensity to consume out of wages',␣
↪default=alpha) # 1
model.param('alpha_2', desc='Propensity to consume out of profits',␣
↪default=alpha_2) # 0.6
model.param('gamma_F', desc='% of undistributed profits', default=gamma_F) #␣
↪0.4
model.param('gamma_u', desc='Adjustment parameter for the marginal propensity␣
↪to invest', default=gamma_u) # 0.01
model.param('omega', desc='Wage-share', default = omega)
model.param('rm', desc='Interest rates on money deposits', default=rm) # 0.02
model.param('spread_l', desc='Spread for loans', default=spread_l) # 0.01
model.param('spread_mo', desc='Spread for mortgages', default=spread_mo) # 0.
↪005
model.param('un', desc='Normal capacity utilization ratio', default=un) # 0.8
model.param('v', desc='Capitl-Output ratio', default=v) # 2.5
model.param('phi_0', desc='Autonomous housing investment component',default =␣
↪phi_0)
model.param('phi_1', desc='Housing investment sensitivity to own interest␣
↪rate', default = phi_1)
model.param('phparam', desc='ph parameter', default = phparam)
model.param('infla', desc='infla value', default = infla)
model.param('gZn', desc='Growth rate new autonomous expenditure',
            default=phi_0 - phi_1*data['Taxa Própria'].iloc[-1])
```

```
# General equations
model.add('Y = C + I_t') # Eq1
model.add('C = C_w + C_k')
model.add('I_t = I_f + I_h') # Eq2
model.add('Yk = K_f(-1)/v') # Eq 4
model.add('u = Y/Yk') # Eq 5
model.add('W = omega*Y') # Eq 6
model.add('gk = h*u/v') # Eq 7
model.add('Knom = K_HD*ph + K_f')
model.add('K = K_HD + K_f')
model.add('Z = I_h') # Eq 9

# Household equations
model.add('YD_w = W') # Eq 10 ################################### Changes #################################################
model.add('YD_k = FD + rm*M(-1) - rmo*MO(-1)') # Eq 10 ################################### Changes #################################################
model.add('Sh_w = YD_w - C_w') # Eq 11 ################################### Changes #################################################
model.add('Sh_k = YD_k - C_k') # Eq 11 ################################### Changes #################################################
model.add('d(MO) = I_h') # Eq 12
model.add('C_k = alpha_2*FD') # Eq 14 ################################### Changes #################################################
model.add('C_w = alpha*W') # Eq 14 ################################### Changes #################################################
model.add('V_h =  M  + K_HD*ph - MO') # Eq 15
model.add('NFW_h = Sh_k - I_h') # Eq 16 ################################### Changes #################################################

# Firms
model.add('d(Lf) = I_f - FU') # Eq 15
model.add('FT = (1-omega)*Y') # Eq 16
model.add('Fn = FT -rl*Lf(-1)')
model.add('FU = gamma_F*(Fn)') # Eq 17
model.add('FD = (1 - gamma_F)*(Fn)') # Eq 18
model.add('I_f = h*Y') # Eq 19
model.add('d(K_f) = I_f') # 20
model.add('h = h(-1)*gamma_u*(u-un) + h(-1)') # Eq 21 # Version without corridor
model.add('V_f = K_f - Lf') # Eq 22
model.add('NFW_f = FU - I_f') # Eq 23
```

```
  # Banks
  model.add('L = Lf')
  model.add('d(M) = d(L) + d(MO)') # Eq 24
  model.add('rmo = rm + spread_mo') # Eq 25
  model.add('rl = rm + spread_l') # Eq 26
  model.add('V_b = L + MO - M') # Eq 27
  model.add('NFW_b = rl*L(-1) + rmo*MO(-1) - rm*M(-1)') # Eq 28

  # Residential investment
  model.add('K_HS = K_HD') # Eq 29
  model.add('Is = I_h')
  model.add('d(K_HD) = I_h') # Eq 30
  model.add('I_h = (1+g_Z)*I_h(-1)') # Eq 31
  model.add('K_k = K_HD/K') #################### Changes␣
 ↪##################################################
  model.add('ph =(1+infla)*ph(-1)')########################### Changes␣
 ↪##############################################
  model.add(
        'own = ((1+rmo)/(1+infla)) -1'
    )  ################################### Changes ############################
  model.add('g_Z = phi_0 - phi_1*own') ###################################␣
 ↪Changes ################################

  model.add('Residual = d(M) - Sh_k - Sh_w')
  return model

t_check = 100
print('Evaluating consistenty at time = {}'.format(t_check))

test = model()
SolveSFC(test, time=t_check, table = False)
evaldf = pd.DataFrame({
    'Households' : test.evaluate('M - MO + K_HD*ph - V_h'),␣
 ↪############################# Changes ######################
    'Firms' : test.evaluate('K_f - Lf - V_f'),
    'Banks' : test.evaluate('L + MO - M - V_b'),
    'Mortgages' : test.evaluate('MO(-1) - MO(-2) - I_h(-1)'),
    'Financial assets' : test.evaluate('d(MO) + d(L) - d(M)'),
    '[Total Wealth - K]' : test.evaluate('V_f + V_h + V_b - K'),
    "Firm's Funds" : test.evaluate('Lf(-1) - Lf(-2) - I_f(-1) + FU(-1)'),
    "Housing" : test.evaluate('K_HD - K_HS'),
    "Workers saving": test.evaluate('YD_w - C_w'),
    "Capitalist saving": test.evaluate('YD_k - C_k - Sh_k'),
    "Flow check" : test.evaluate('NFW_h + NFW_f + NFW_b'),
    "Investment" : test.evaluate('I_t - I_h - I_f'),
    "Profits" : test.evaluate('FT(-1) - rl*Lf(-2) - FU(-1) - FD(-1)'),
    "Wages" : test.evaluate('W - omega*Y'),
```

```
    "Residual" : test.evaluate('Residual')
}, index = ['Sum'])
evaldf = evaldf.transpose()
evaldf.round(5)
```

```
Evaluating consistenty at time = 100
```

[7]:

	Sum
Households	-0.00009
Firms	0.00000
Banks	0.00000
Mortgages	0.00000
Financial assets	0.00000
[Total Wealth - K]	0.00013
Firm's Funds	0.00008
Housing	0.00000
Workers saving	0.00000
Capitalist saving	-0.00000
Flow check	-0.00022
Investment	-0.00004
Profits	0.00000
Wages	0.00000
Residual	-0.19976

7 Solving

[6]:
```
base = model()
df = SolveSFC(base, time=1000)
df.transpose()
```

[6]:

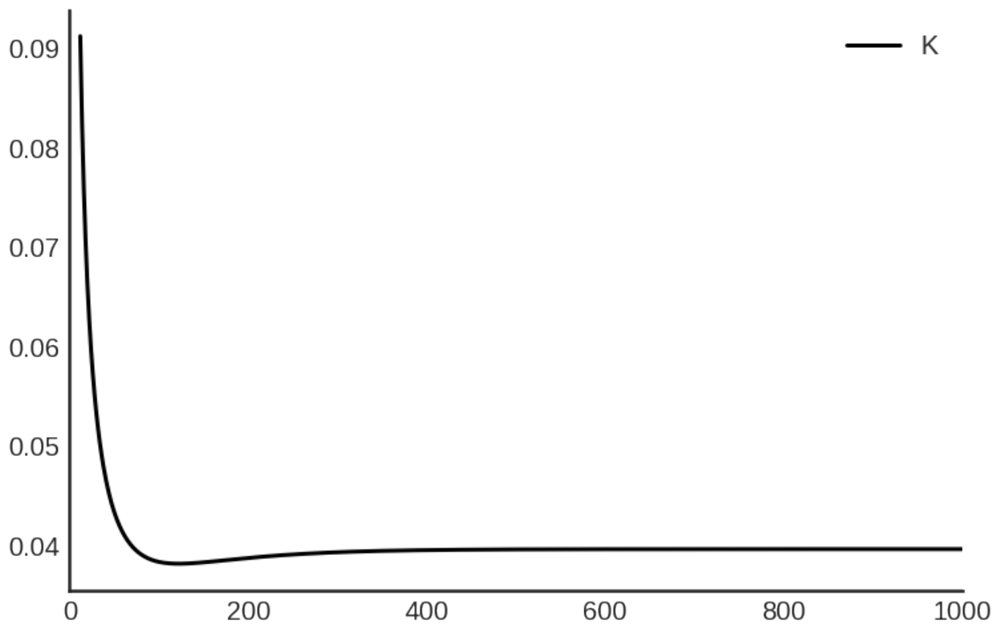
```
                0           1           2           3           4           5   \
C             0.0   87.967573   91.449946   95.048891   98.770855  102.621943
C_w           0.0   82.213358   85.416921   88.726708   92.148368   95.687285
C_k           0.0    5.754891    6.033472    6.322641    6.622953    6.934951
FD            0.0   19.183117   20.111672   21.075570   22.076614   23.116568
Fn            0.0  191.831169  201.116719  210.755697  220.766137  231.165679
...           ...         ...         ...         ...         ...         ...
_h__1         0.3    0.300000    0.299655    0.299229    0.298731    0.298168
_L__1         0.0    0.000000  -90.528459 -186.335588 -287.663933 -394.767401
_K_HD__1    500.0  500.000000  603.960000  712.036816  824.393474  941.199455
_I_h__1     100.0  100.000000  103.960000  108.076816  112.356658  116.805982
_ph__1        1.0    1.000000    1.000000    1.000000    1.000000    1.000000

                   6           7           8           9    ...  \
C         106.606201  110.729969  114.999022  119.419298   ...
C_w        99.347450  103.134458  107.053542  111.110087   ...
```

```
C_k          7.259049     7.595814     7.945787     8.309523  ...
FD          24.196894    25.319448    26.486024    27.698478  ...
Fn         241.968943   253.194477   264.860241   276.984779  ...
...               ...          ...          ...          ...  ...
_h__1        0.297547     0.296875     0.296156     0.295396  ...
_L__1     -507.911275  -627.370464  -753.432140  -886.395504  ...
_K_HD__1  1062.630954  1188.871140  1320.110437  1456.546810  ...
_I_h__1    121.431498   126.240186   131.239297   136.436373  ...
_ph__1       1.000000     1.000000     1.000000     1.000000  ...

                   991           992           993           994   \
C         3.208141e+18  3.335184e+18  3.467257e+18  3.604560e+18
C_w       2.872427e+18  2.986175e+18  3.104427e+18  3.227363e+18
C_k       3.357145e+17  3.490088e+17  3.628295e+17  3.771975e+17
FD        1.119048e+18  1.163363e+18  1.209432e+18  1.257325e+18
Fn        1.119048e+19  1.163363e+19  1.209432e+19  1.257325e+19
...                ...           ...           ...           ...
_h__1     1.237514e-01  1.237514e-01  1.237514e-01  1.237514e-01
_L__1    -2.244076e+20 -2.332942e+20 -2.425326e+20 -2.521369e+20
_K_HD__1  1.308517e+20  1.360334e+20  1.414203e+20  1.470205e+20
_I_h__1   4.984345e+18  5.181725e+18  5.386922e+18  5.600244e+18
_ph__1    1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00

                   995           996           997           998   \
C         3.747301e+18  3.895694e+18  4.049963e+18  4.210341e+18
C_w       3.355166e+18  3.488031e+18  3.626156e+18  3.769752e+18
C_k       3.921346e+17  4.076631e+17  4.238065e+17  4.405893e+17
FD        1.307115e+18  1.358877e+18  1.412688e+18  1.468631e+18
Fn        1.307115e+19  1.358877e+19  1.412688e+19  1.468631e+19
...                ...           ...           ...           ...
_h__1     1.237514e-01  1.237514e-01  1.237513e-01  1.237513e-01
_L__1    -2.621215e+20 -2.725016e+20 -2.832926e+20 -2.945110e+20
_K_HD__1  1.528426e+20  1.588951e+20  1.651874e+20  1.717288e+20
_I_h__1   5.822013e+18  6.052565e+18  6.292247e+18  6.541420e+18
_ph__1    1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00

                   999           1000
C         4.377071e+18  4.550403e+18
C_w       3.919034e+18  4.074228e+18
C_k       4.580366e+17  4.761748e+17
FD        1.526789e+18  1.587249e+18
Fn        1.526789e+19  1.587249e+19
...                ...           ...
_h__1     1.237513e-01  1.237513e-01
_L__1    -3.061737e+20 -3.182981e+20
_K_HD__1  1.785292e+20  1.855990e+20
_I_h__1   6.800460e+18  7.069758e+18
```

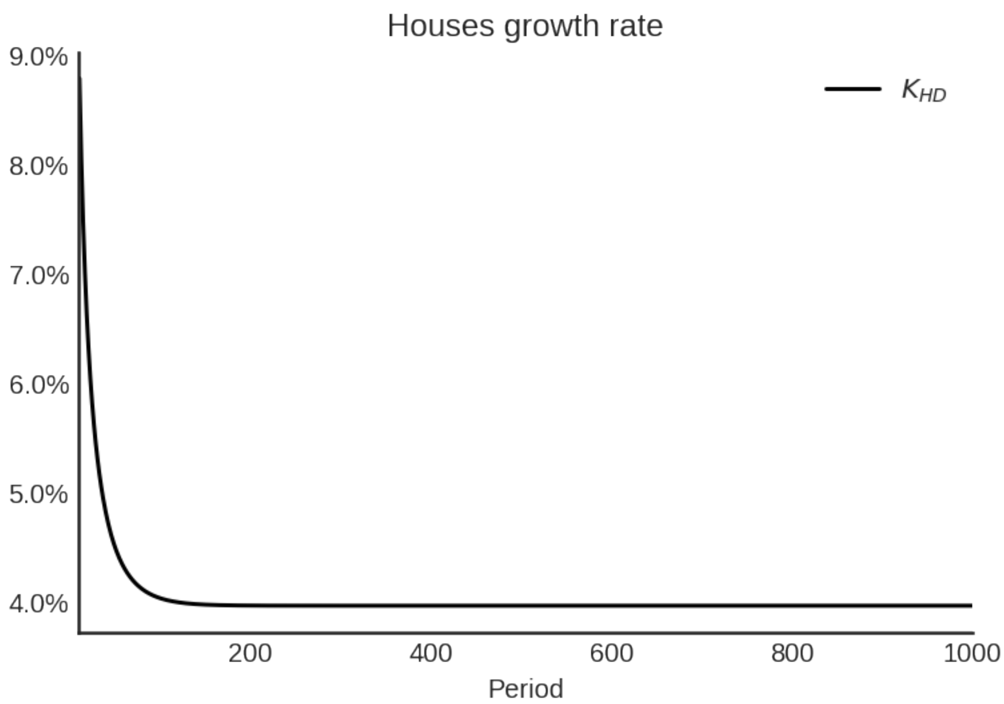
```
_ph__1     1.000000e+00  1.000000e+00

[67 rows x 1001 columns]
```

8 Baseline Plots

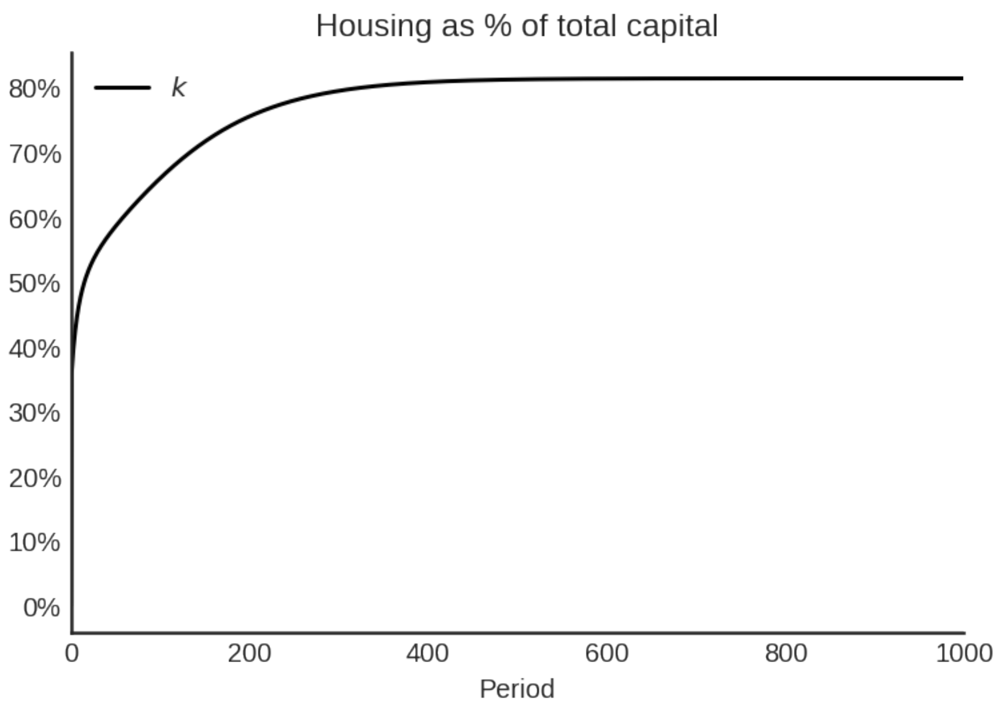
[7]:
```python
fig, ax = plt.subplots()

df[["K"]].pct_change().rolling(12).mean().plot(ax=ax, color='black')
sns.despine()
plt.show()
```



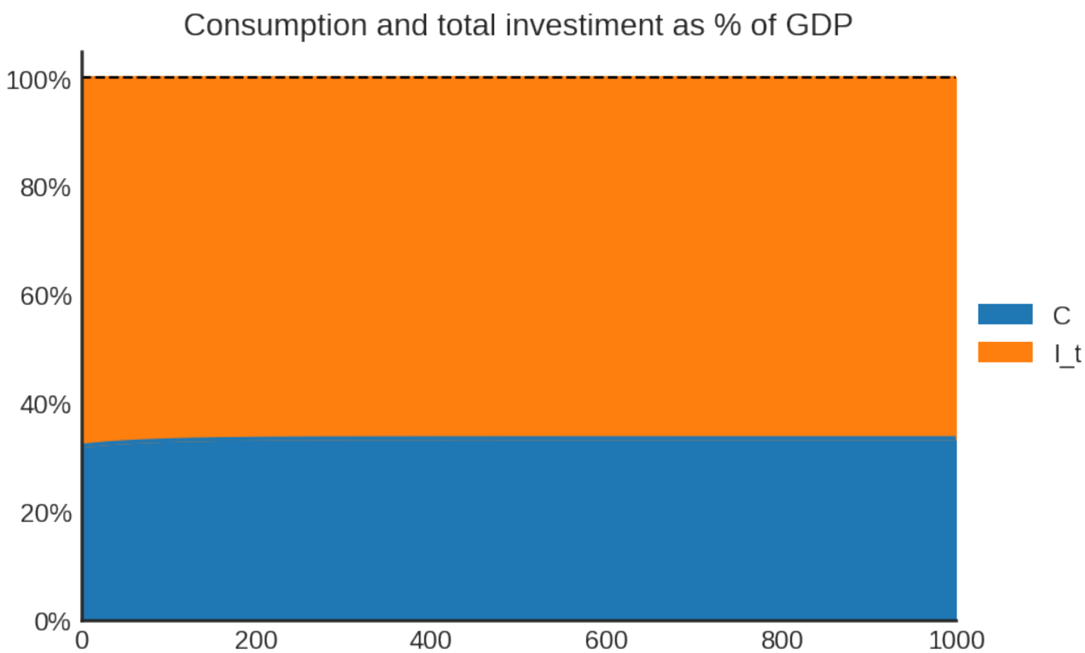
[8]:
```python
ax = df['K_HD'][10:].pct_change().plot(color="black",
                                       title="Houses growth rate",
                                       label="$K_{HD}$",
                                       legend=True)
ax.set_yticklabels(['{:,.1%}'.format(x) for x in ax.get_yticks()])
plt.xlabel("Period")
sns.despine()
plt.show()
```



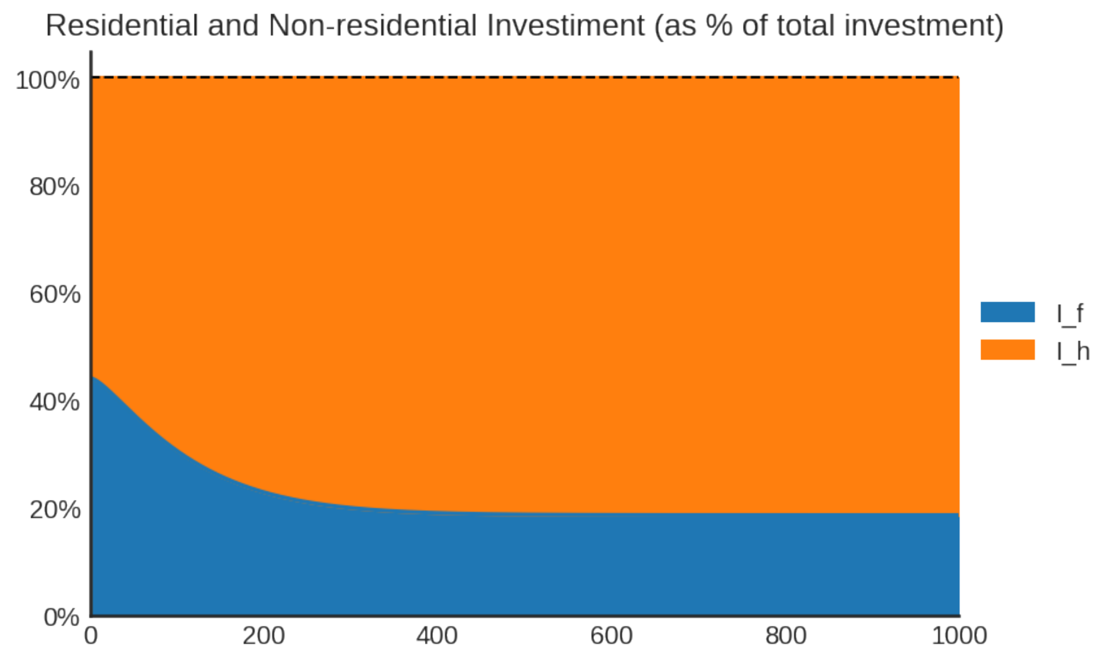
[9]:
```
ax = df['K_k'].plot(color = "black", title = "Housing as % of total capital",␣
 ↪label = "$k$", legend = True)
ax.set_yticklabels(['{:,.0%}'.format(x) for x in ax.get_yticks()])
plt.xlabel("Period")
sns.despine()
plt.show()
```



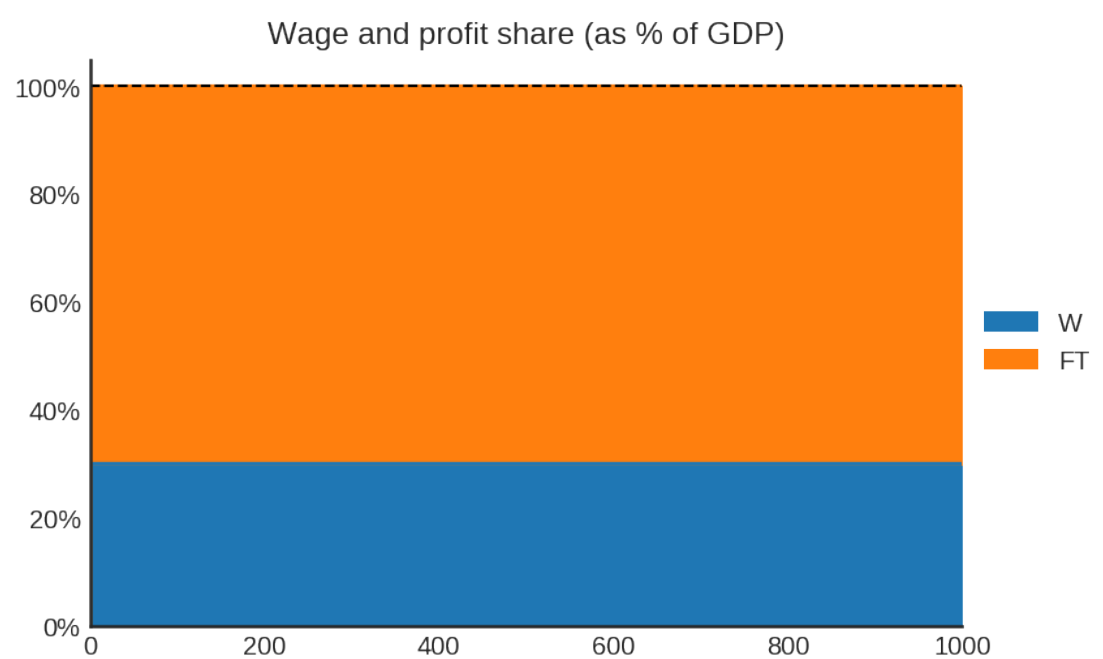
[10]:
```
ax = df[['C', 'I_t']].apply(lambda x: x/df['Y']).plot(kind = 'area', stacked = True, legend = True, title = "Consumption and total investiment as % of GDP")
ax.axhline(y=1, color = "black", ls = "--", lw=1)
ax.legend(loc='center left', bbox_to_anchor=(1, 0.5))
ax.set_yticklabels(['{:,.0%}'.format(x) for x in ax.get_yticks()])
sns.despine()
plt.show()
```



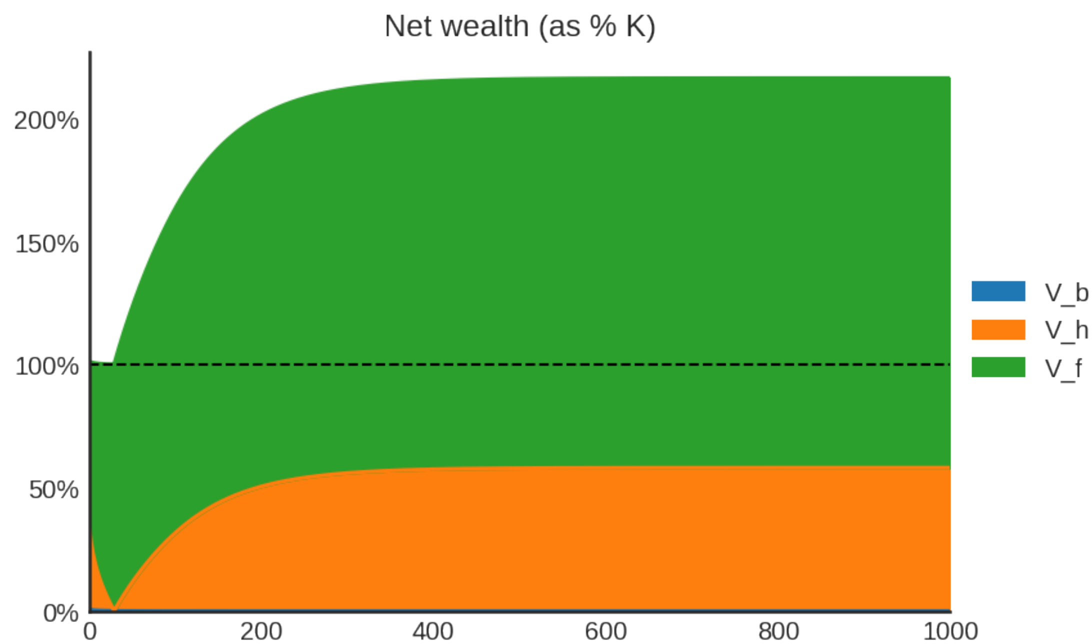
[11]:
```
ax = df[['I_f', 'I_h']].apply(lambda x: x/df['I_t']).plot(kind = 'area',␣
↪stacked = True, legend = True, title = "Residential and Non-residential␣
↪Investiment (as % of total investment)")
ax.axhline(y=1, color = "black", ls = "--", lw=1)
ax.legend(loc='center left', bbox_to_anchor=(1, 0.5))
ax.set_yticklabels(['{:,.0%}'.format(x) for x in ax.get_yticks()])
sns.despine()
plt.show()
```



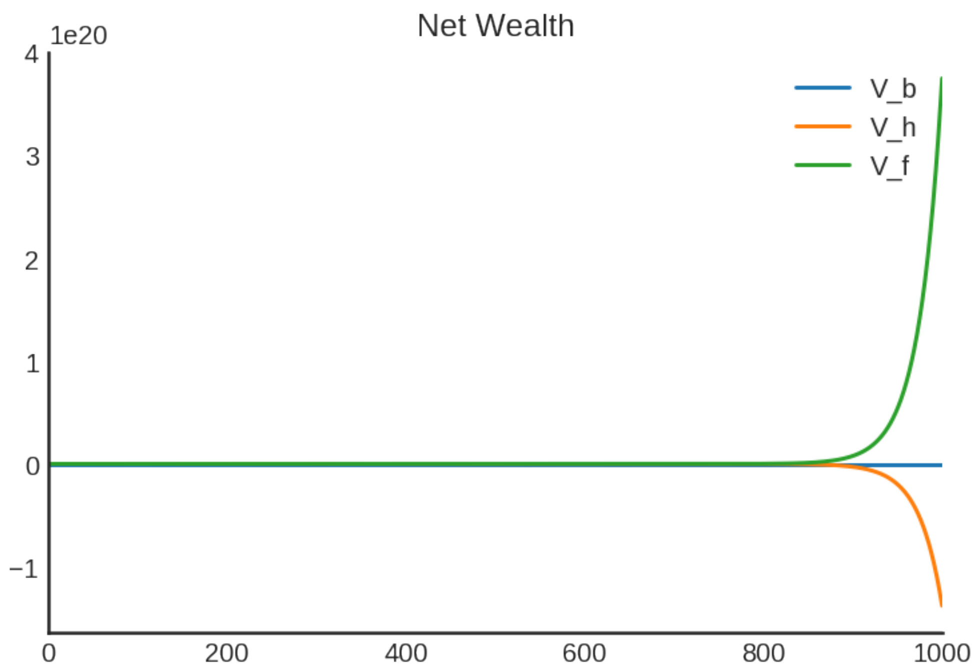
[12]:
```
ax = df[['W', 'FT']].apply(lambda x: x/df['Y']).plot(kind = 'area', stacked =
↪True, legend = True, title = "Wage and profit share (as % of GDP)")
ax.axhline(y=1, color = "black", ls = "--", lw=1)
ax.legend(loc='center left', bbox_to_anchor=(1, 0.5))
ax.set_yticklabels(['{:,.0%}'.format(x) for x in ax.get_yticks()])
sns.despine()
plt.show()
```



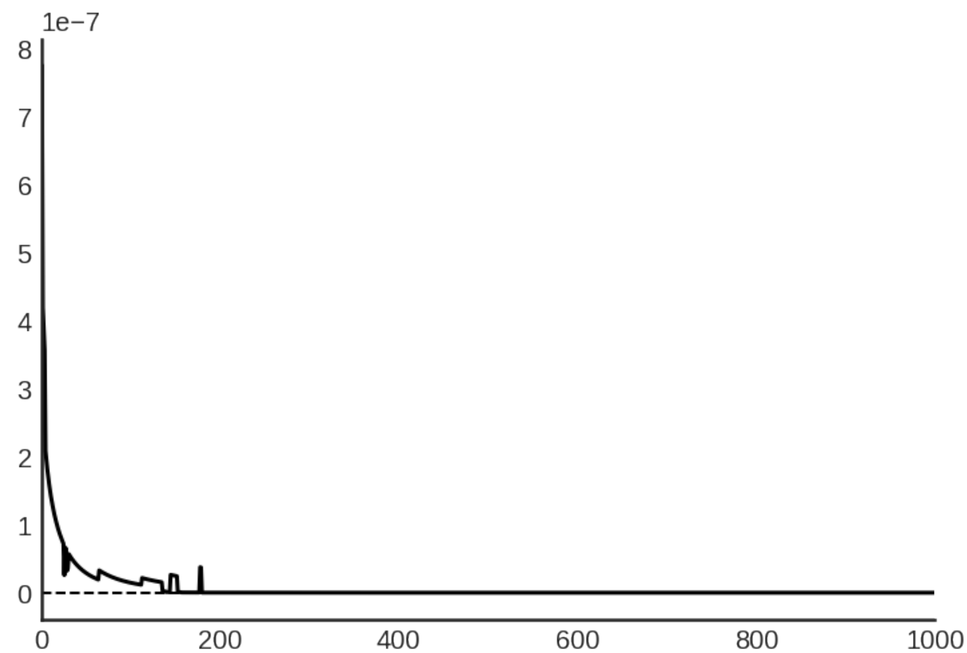
[13]:
```
ax = df[['V_b', 'V_h', 'V_f']].apply(lambda x: np.abs(x)/df['K']).plot(kind =␣
↪'area', stacked = True, title = "Net wealth (as % K)")
ax.set_yticklabels(['{:,.0%}'.format(x) for x in ax.get_yticks()])
ax.axhline(y=1, color = "black", ls = "--", lw=1)
ax.legend(loc='center left', bbox_to_anchor=(1, 0.5))
sns.despine()
plt.show()
```



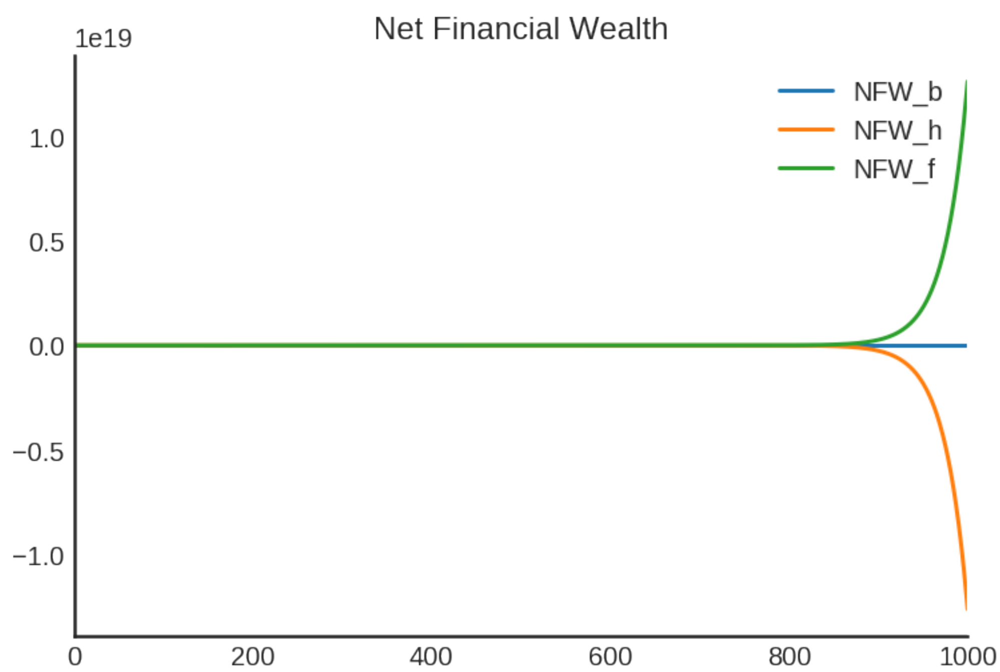
[14]:
```
df[["V_b", "V_h", "V_f"]].plot(title = "Net Wealth")
sns.despine()
plt.show()
```



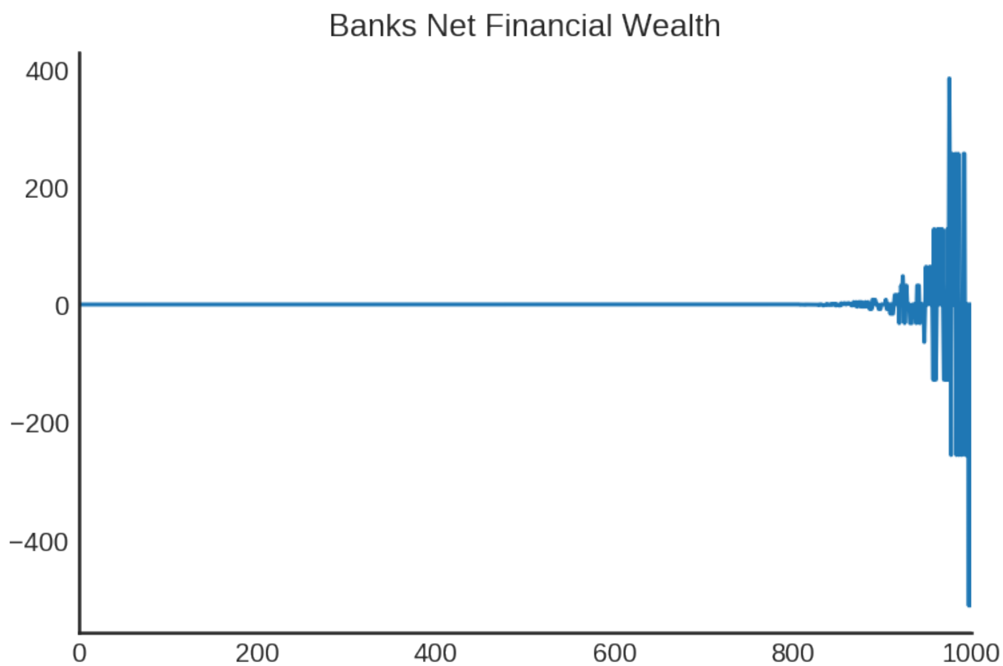
[15]:
```
ax = ((df['V_b'] + df['V_f'] + df['V_h'] - df['K'])/df['K']).
 ↪plot(label='Consistensy check', color='black')
ax.axhline(y=0, color = "black", ls = "--", lw=1)
sns.despine()
plt.show()
```



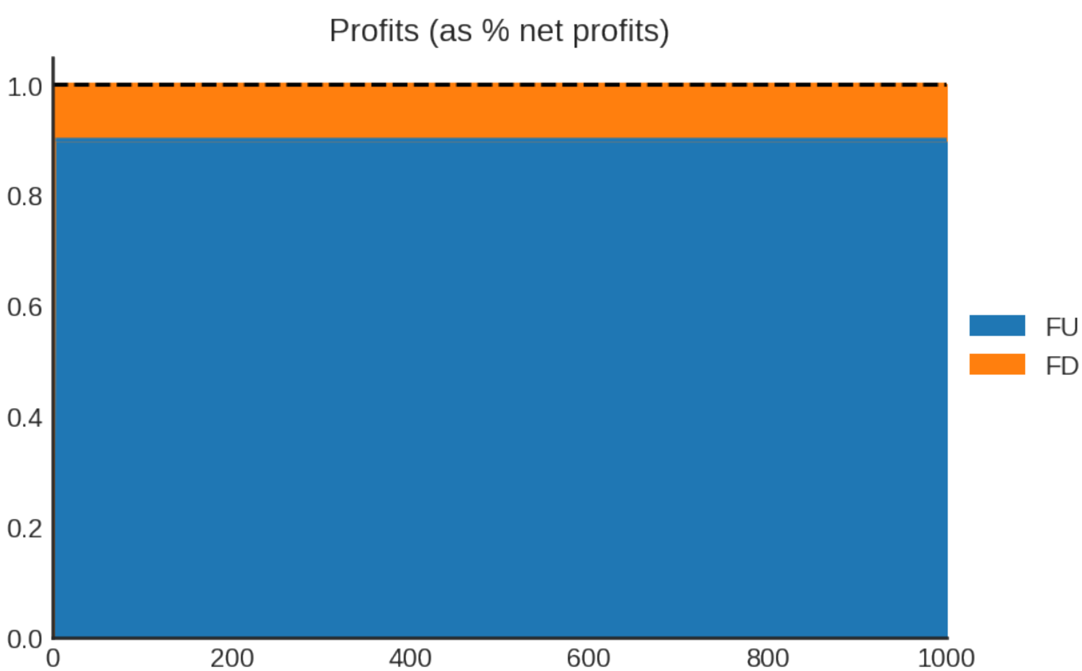
[16]:
```
df[["NFW_b", "NFW_h", "NFW_f"]].plot(title = "Net Financial Wealth")
sns.despine()
plt.show()
```



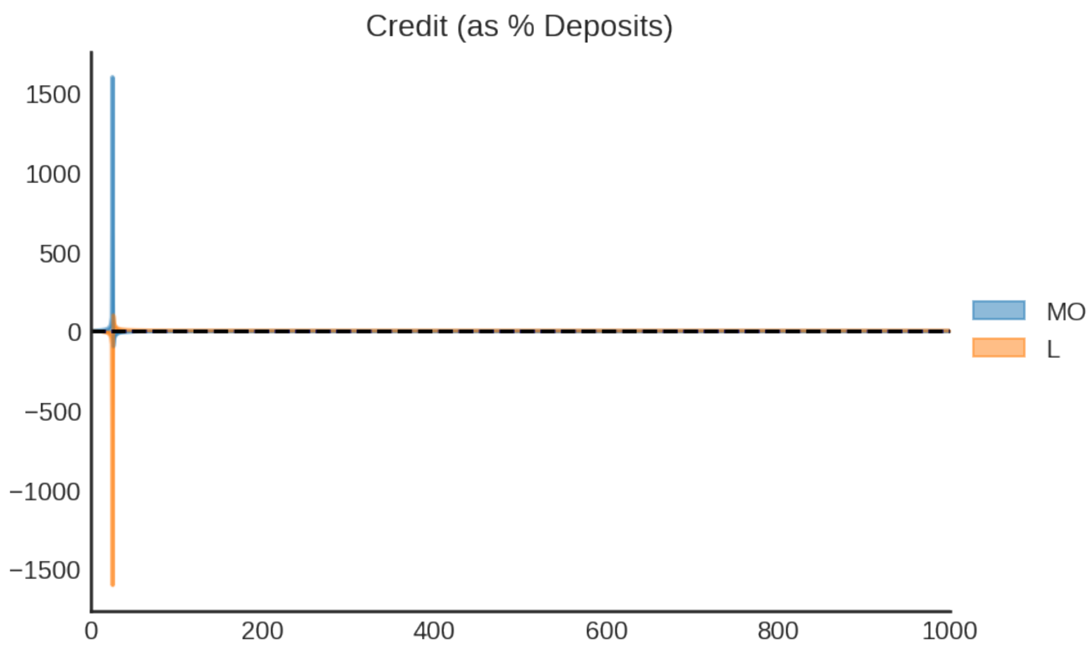
[17]:
```
df['NFW_b'].plot(title = "Banks Net Financial Wealth")
sns.despine()
plt.show()
```



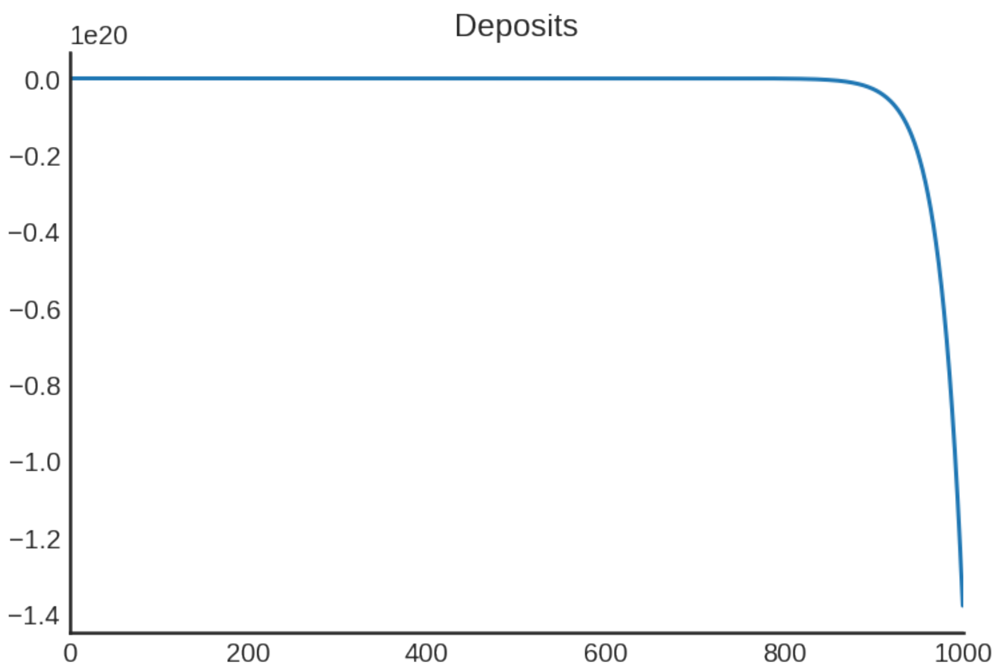
[18]:
```
ax = df[["FU", "FD"]].apply(lambda x: x/df["Fn"]).plot(kind = "area", stacked =␣
↪True, title = "Profits (as % net profits)")
ax.legend(loc='center left', bbox_to_anchor=(1, 0.5))
ax.axhline(y = 1, color = "black", ls = "--")
sns.despine()
plt.show()
```



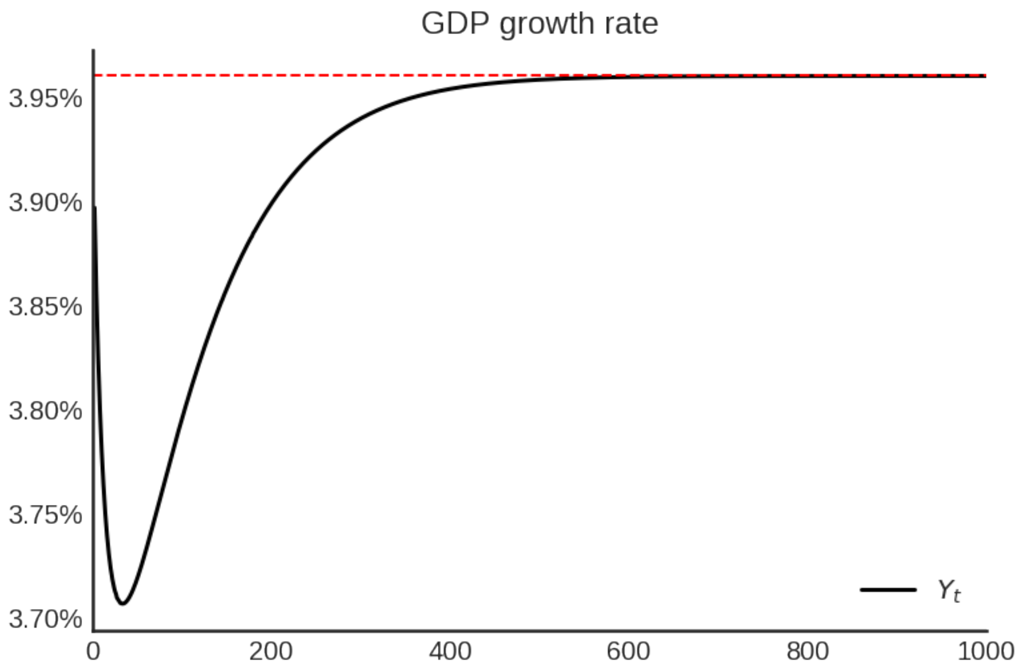
[19]:
```
ax = df[["MO", "L"]].apply(lambda x: x/df["M"]).plot(kind = "area",stacked =␣
↪False ,title = "Credit (as % Deposits)")
ax.legend(loc='center left', bbox_to_anchor=(1, 0.5))
ax.axhline(y = 1, color = "blue", ls = "--")
ax.axhline(y = 0, color = "black", ls = "--")
sns.despine()
plt.show()
```



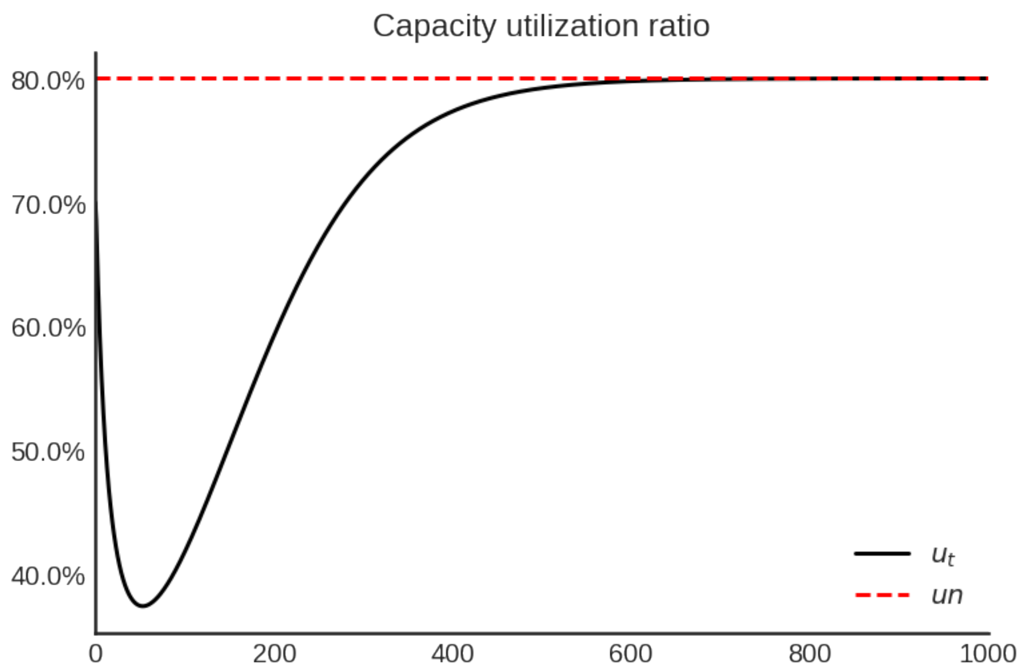
[20]:
```
df["M"].plot(title = "Deposits")
sns.despine()
plt.show()
```



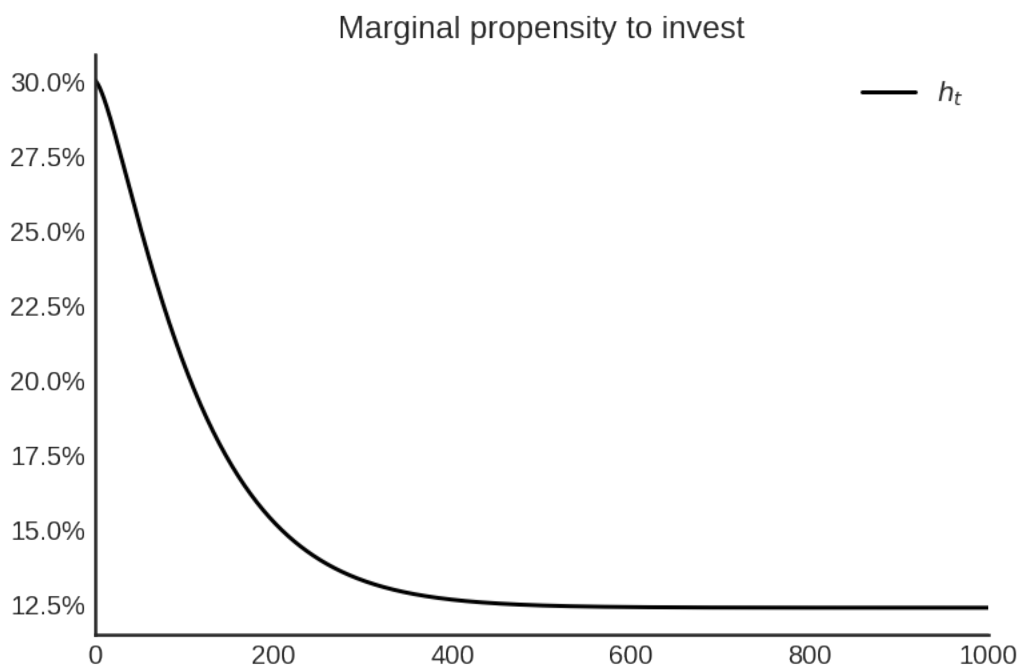
[21]:
```
ax = df['Y'].pct_change().plot(color = "black", title = "GDP growth rate",␣
 ↪label = "$Y_t$", legend = True)
ax.set_yticklabels(['{:,.2%}'.format(x) for x in ax.get_yticks()])
ax.axhline(y=df["g_Z"].iloc[-1], color = "red", ls = "--", lw=1, label =␣
 ↪"Autonomous growth rate")
sns.despine()
plt.show()
```



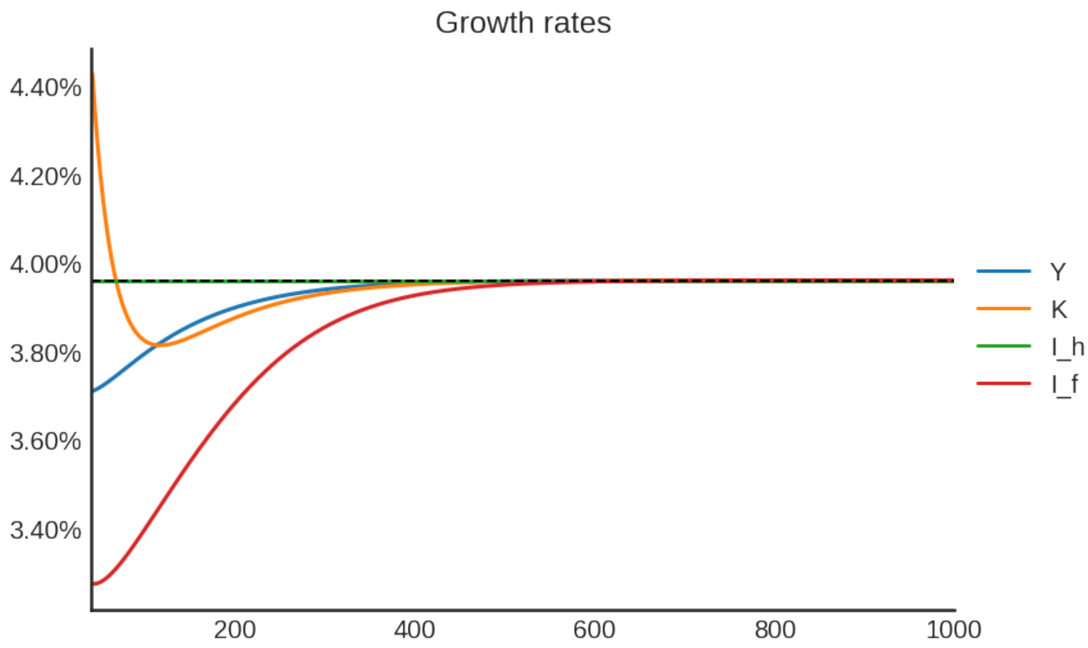
[22]:
```
ax = df['u'].plot(color = "black", title = "Capacity utilization ratio", label
 ↪= "$u_t$", legend = True)
ax.set_yticklabels(['{:,.1%}'.format(x) for x in ax.get_yticks()])
ax = df['un'].plot(color = "red", ls="--",  title = "Capacity utilization
 ↪ratio", label = "$un$", legend = True)
ax.set_yticklabels(['{:,.1%}'.format(x) for x in ax.get_yticks()])
sns.despine()
plt.show()
```



[23]:
```
ax = df['h'].plot(color = "black", ls="-",  title = "Marginal propensity to␣
↪invest", label = "$h_t$", legend = True)
ax.set_yticklabels(['{:,.1%}'.format(x) for x in ax.get_yticks()])
sns.despine()
plt.show()
```

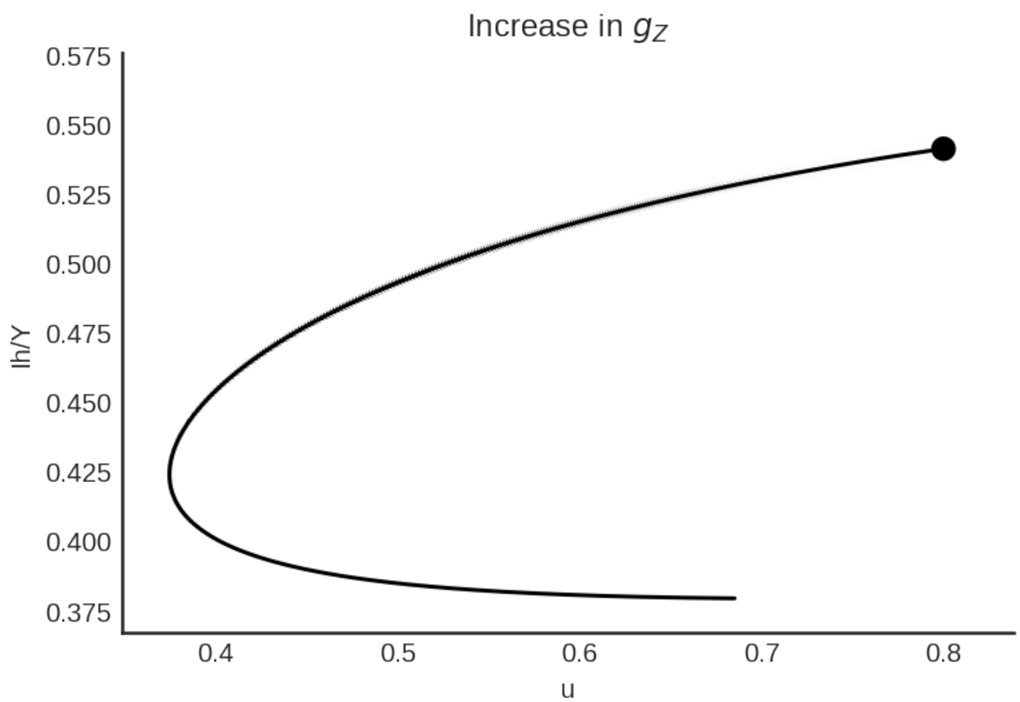


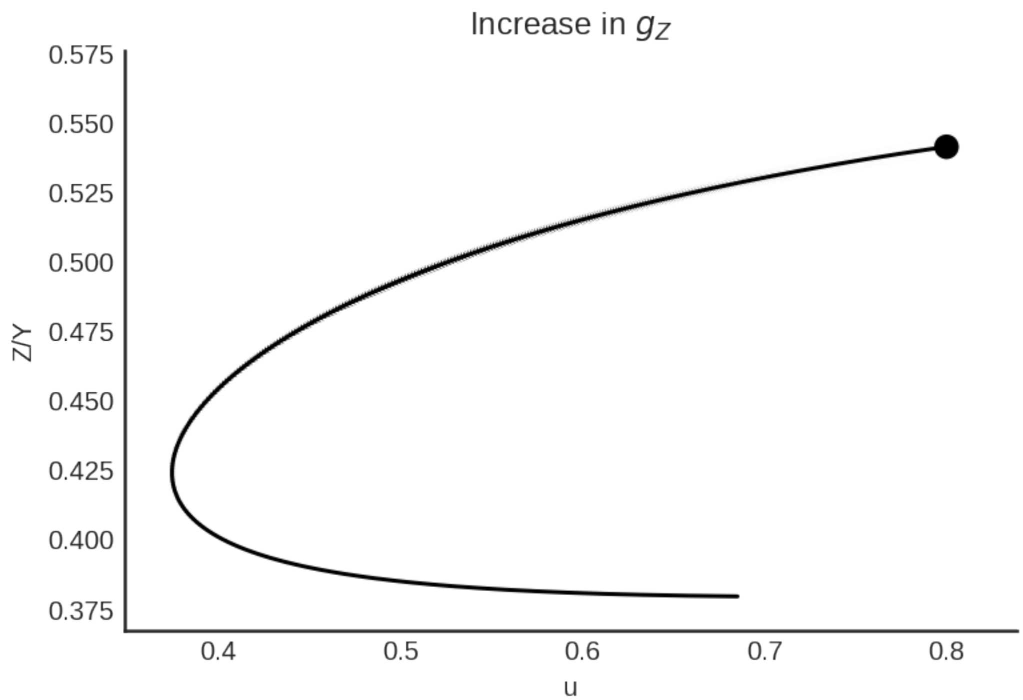
[24]:
```
ax = df[["Y", "K", "I_h", "I_f"]][40:].pct_change().plot(title = "Growth rates")
ax.set_yticklabels(['{:,.2%}'.format(x) for x in ax.get_yticks()])
ax.axhline(y=df["g_Z"].iloc[-1], color = "black", ls = "--", lw=1)
ax.legend(loc='center left', bbox_to_anchor=(1, 0.5))
sns.despine()
plt.show()
```



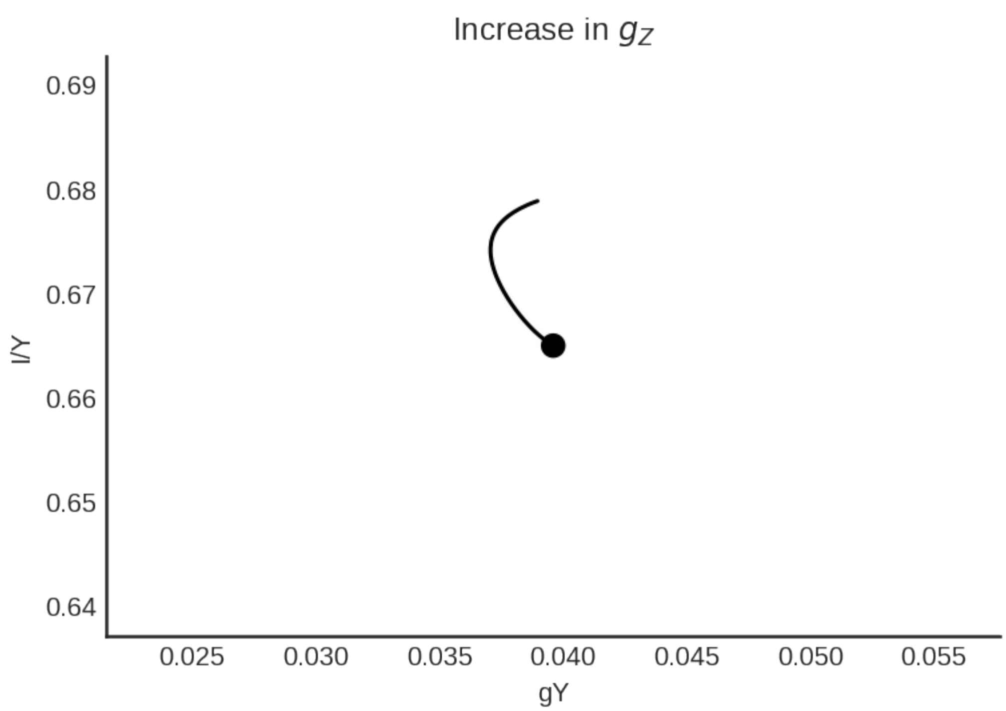
[25]:
```
df["TIME"] = [i+1 for i in range(len(df.index))]
df["Ih/Y"] = df["I_h"]/df["Y"]
df["I/Y"] = df["I_t"]/df["Y"]
df["Z/Y"] = df["Z"]/df["Y"]
df["gY"] = df["Y"].pct_change()

sns.scatterplot(y = 'Ih/Y', x='u', data=df, size="TIME", sizes = (1,100), color␣
↪= 'black', legend=False)
sns.lineplot(y = 'Ih/Y', x='u', data=df, sort=False, color = 'black')
plt.title('Increase in $g_Z$')
sns.despine()
plt.show()

sns.scatterplot(y = 'Z/Y', x='u', data=df, size="TIME", sizes = (1,100), color␣
↪= 'black', legend=False)
sns.lineplot(y = 'Z/Y', x='u', data=df, sort=False, color = 'black')
plt.title('Increase in $g_Z$')
```

```
sns.despine()
plt.show()

sns.scatterplot(y = 'I/Y', x='gY', data=df, size="TIME", sizes = (1,100), color␣
↪= 'black', legend=False)
sns.lineplot(y = 'I/Y', x='gY', data=df, sort=False, color = 'black')
plt.title('Increase in $g_Z$')
sns.despine()
plt.show()
```





Increase in g_Z
0.575
0.550
0.525
0.500
0.475
0.450
0.425
0.400
0.375
Z/Y
0.4
0.5
0.6
0.7
0.8
u



Increase in g_Z
0.69
0.68
0.67
0.66
0.65
0.64
I/Y
0.025
0.030
0.035
0.040
0.045
0.050
0.055
gY

9 Shocks

[26]:
```python
def plot_shock(filename, shock):
    """
    This function plots some selected variables

    filename: name to save the plot (str)
    shock: df returned by ShockModel function
    """
    sns.set_context('paper')
    fig, ax = plt.subplots(2,2)

    shock[["Y"]].pct_change().plot(title = "Growth rates", ax = ax[0,0],
                                                       ls = ('--'),
                                                       )
    shock[["K"]].pct_change().plot(title = "Growth rates", ax = ax[0,0],
                                                       ls = (':'),
                                                       )
    shock[["I_h"]].pct_change().plot(title = "Growth rates", ax = ax[0,0],
                                                       ls = ('-'),
                                                       )
    shock[["I_f"]].pct_change().plot(title = "Growth rates", ax = ax[0,0],
                                                       ls = ('-.'),
                                                       )
    ax[0,0].axhline(y=shock["g_Z"].iloc[-1], color = "black", ls = "--", lw=1)
    #ax[0,0].set_yticklabels(['{:,.1%}'.format(x) for x in ax[0,0].get_yticks()])
    ax[0,0].legend(loc='upper center', bbox_to_anchor=(0.5, -0.06),
                   labels = ["$Y$", "$K$", "$I_h$", "$I_f$"],
              fancybox=True, shadow=True, ncol=2)
    ax[0,0].ticklabel_format(useOffset=False)

    shock['K_k'].plot(color = "darkred", title = "Houses \nas % of Total capital", label = "$\k$", legend = False, ax = ax[0,1], lw = 3, )
    ax[0,1].axhline(y = df['K_k'].iloc[-1], ls ='--', color = "gray")
    ax[0,1].ticklabel_format(useOffset=False)


    shock['u'].plot(title = 'Capacity utilization ratio', ax=ax[1,0], legend = False, color = "darkred", lw = 3, )
    ax[1,0].axhline(y = shock['un'].iloc[-1], ls ='--', color = "gray")
    #ax[1,0].set_yticklabels(['{:,.2%}'.format(x) for x in ax[1,0].get_yticks()])
    ax[1,0].ticklabel_format(useOffset=False)

    shock['h'].plot(title = 'Marginal propensity to invest', ax=ax[1,1], legend = False, color = "darkred", lw = 3, )
```

```python
    ax[1,1].axhline(y = df['h'].iloc[-1], ls ='--', color = "gray")
    ax[1,1].ticklabel_format(useOffset=False)

    sns.despine()
    plt.tight_layout(rect=[0, 0.03, 1, 0.95])
    plt.show()

    fig.savefig(filename, dpi = 300)
```

9.1 Increase in autonomous growth rate (g_z)

[27]:
```python
base = model()
df = SolveSFC(base, time=900)
shock = ShockModel(base_model=base, create_function=model(), variable='phi_0',␣
 ↪increase=0.01, time = 1200)
shock.transpose()
```

[27]:
```
                     0              1              2              3     \
C          9.363219e+16   9.734002e+16   1.011947e+17   1.052020e+17
C_w        8.383412e+16   8.715394e+16   9.060523e+16   9.419319e+16
C_k        9.798072e+15   1.018607e+16   1.058944e+16   1.100878e+16
FD         3.266024e+16   3.395358e+16   3.529814e+16   3.669595e+16
Fn         3.266024e+17   3.395358e+17   3.529814e+17   3.669595e+17
...                 ...            ...            ...            ...
_h__1      1.237546e-01   1.237546e-01   1.237545e-01   1.237545e-01
_L__1     -6.808832e+18  -6.808832e+18  -7.078462e+18  -7.358769e+18
_K_HD__1   3.970222e+18   3.970222e+18   4.127443e+18   4.290890e+18
_I_h__1    1.512320e+17   1.512320e+17   1.572208e+17   1.634467e+17
_ph__1     1.000000e+00   1.000000e+00   1.000000e+00   1.000000e+00

                     4              5              6              7     \
C          1.093680e+17   1.136989e+17   1.182014e+17   1.228821e+17
C_w        9.792323e+16   1.018010e+17   1.058323e+17   1.100232e+17
C_k        1.144473e+16   1.189794e+16   1.236910e+16   1.285892e+16
FD         3.814911e+16   3.965981e+16   4.123033e+16   4.286305e+16
Fn         3.814911e+17   3.965981e+17   4.123033e+17   4.286305e+17
...                 ...            ...            ...            ...
_h__1      1.237544e-01   1.237544e-01   1.237543e-01   1.237543e-01
_L__1     -7.650176e+18  -7.953123e+18  -8.268067e+18  -8.595483e+18
_K_HD__1   4.460809e+18   4.637457e+18   4.821101e+18   5.012016e+18
_I_h__1    1.699192e+17   1.766480e+17   1.836433e+17   1.909156e+17
_ph__1     1.000000e+00   1.000000e+00   1.000000e+00   1.000000e+00

                     8              9     ...          1241           1242  \
C          1.277483e+17   1.328071e+17  ...  7.397364e+42   7.764273e+42
C_w        1.143801e+17   1.189096e+17  ...  6.724489e+42   7.058024e+42
C_k        1.336813e+16   1.389751e+16  ...  6.728746e+41   7.062491e+41
```

```
FD        4.456043e+16  4.632502e+16  ...  2.242915e+42  2.354164e+42
Fn        4.456043e+17  4.632502e+17  ...  2.242915e+43  2.354164e+43
...                ...           ...  ...           ...           ...
_h__1     1.237542e-01  1.237541e-01  ...  1.549999e-01  1.549999e-01
_L__1    -8.935864e+18 -9.289725e+18  ... -3.369339e+44 -3.536458e+44
_K_HD__1  5.210492e+18  5.416827e+18  ...  2.327275e+44  2.442708e+44
_I_h__1   1.984758e+17  2.063355e+17  ...  1.099779e+43  1.154328e+43
_ph__1    1.000000e+00  1.000000e+00  ...  1.000000e+00  1.000000e+00

                  1243          1244          1245          1246  \
C         8.149381e+42  8.553590e+42  8.977848e+42  9.423150e+42
C_w       7.408102e+42  7.775544e+42  8.161211e+42  8.566007e+42
C_k       7.412791e+41  7.780465e+41  8.166376e+41  8.571429e+41
FD        2.470930e+42  2.593488e+42  2.722125e+42  2.857143e+42
Fn        2.470930e+43  2.593488e+43  2.722125e+43  2.857143e+43
...                ...           ...           ...           ...
_h__1     1.549999e-01  1.549999e-01  1.549999e-01  1.549999e-01
_L__1    -3.711866e+44 -3.895975e+44 -4.089215e+44 -4.292040e+44
_K_HD__1  2.563866e+44  2.691034e+44  2.824509e+44  2.964604e+44
_I_h__1   1.211583e+43  1.271677e+43  1.334753e+43  1.400956e+43
_ph__1    1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00

                  1247          1248          1249          1250
C         9.890538e+42  1.038111e+43  1.089601e+43  1.143645e+43
C_w       8.990881e+42  9.436828e+42  9.904895e+42  1.039618e+43
C_k       8.996572e+41  9.442802e+41  9.911164e+41  1.040276e+42
FD        2.998857e+42  3.147601e+42  3.303721e+42  3.467586e+42
Fn        2.998857e+43  3.147601e+43  3.303721e+43  3.467586e+43
...                ...           ...           ...           ...
_h__1     1.549999e-01  1.549999e-01  1.549999e-01  1.549999e-01
_L__1    -4.504925e+44 -4.728369e+44 -4.962897e+44 -5.209056e+44
_K_HD__1  3.111649e+44  3.265987e+44  3.427980e+44  3.598007e+44
_I_h__1   1.470444e+43  1.543378e+43  1.619929e+43  1.700278e+43
_ph__1    1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00

[67 rows x 1251 columns]
```

[28]:

```
summary = SummaryShock(shock)
summary
```

[28]:

```
                     0         Shock             1             2  \
C         6.527354e+17  6.846897e+17  7.182326e+17  7.534437e+17
C_w       5.844302e+17  6.132802e+17  6.435693e+17  6.753698e+17
C_k       6.830513e+16  7.140947e+16  7.466328e+16  7.807393e+16
FD        2.276838e+17  2.380316e+17  2.488776e+17  2.602464e+17
Fn        2.276838e+18  2.380316e+18  2.488776e+18  2.602464e+18
...                ...           ...           ...           ...
_h__1     1.237525e-01  1.237524e-01  1.237617e-01  1.237800e-01
```

```
_L__1     -4.565836e+19 -4.746643e+19 -4.935571e+19 -5.133008e+19
_K_HD__1   2.662331e+19  2.767760e+19  2.878417e+19  2.994563e+19
_I_h__1    1.014124e+18  1.054283e+18  1.106576e+18  1.161462e+18
_ph__1     1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00

                      3           t-1             t    difference
C          7.904064e+17  1.089601e+43  1.143645e+43  5.404422e+41
C_w        7.087572e+17  9.904895e+42  1.039618e+43  4.912828e+41
C_k        8.164915e+16  9.911164e+41  1.040276e+42  4.915938e+40
FD         2.721638e+17  3.303721e+42  3.467586e+42  1.638646e+41
Fn         2.721638e+18  3.303721e+43  3.467586e+43  1.638646e+42
...                 ...           ...           ...           ...
_h__1      1.238071e-01  1.549999e-01  1.549999e-01  8.355082e-10
_L__1     -5.339358e+19 -4.962897e+44 -5.209056e+44 -2.461597e+43
_K_HD__1   3.116470e+19  3.427980e+44  3.598007e+44  1.700278e+43
_I_h__1    1.219070e+18  1.619929e+43  1.700278e+43  8.034850e+41
_ph__1     1.000000e+00  1.000000e+00  1.000000e+00  0.000000e+00

[67 rows x 8 columns]
```

[29]:

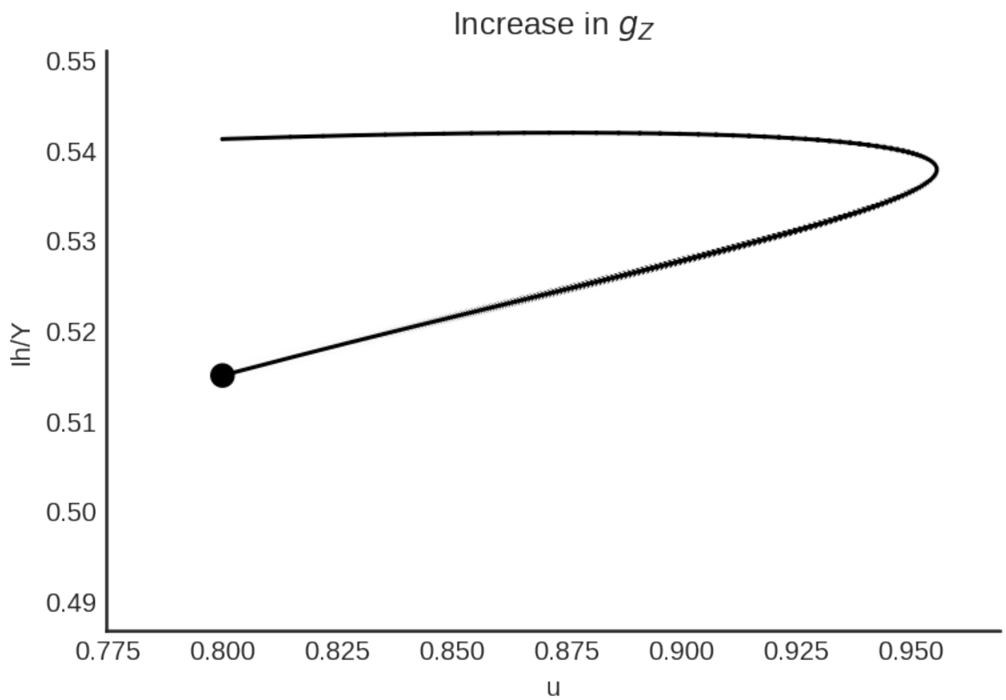
```python
shock1 = shock.round(decimals = 5).tail(1).transpose().loc['alpha':,:]
shock1.columns = ['$\Delta \phi_0$']
shock1
```

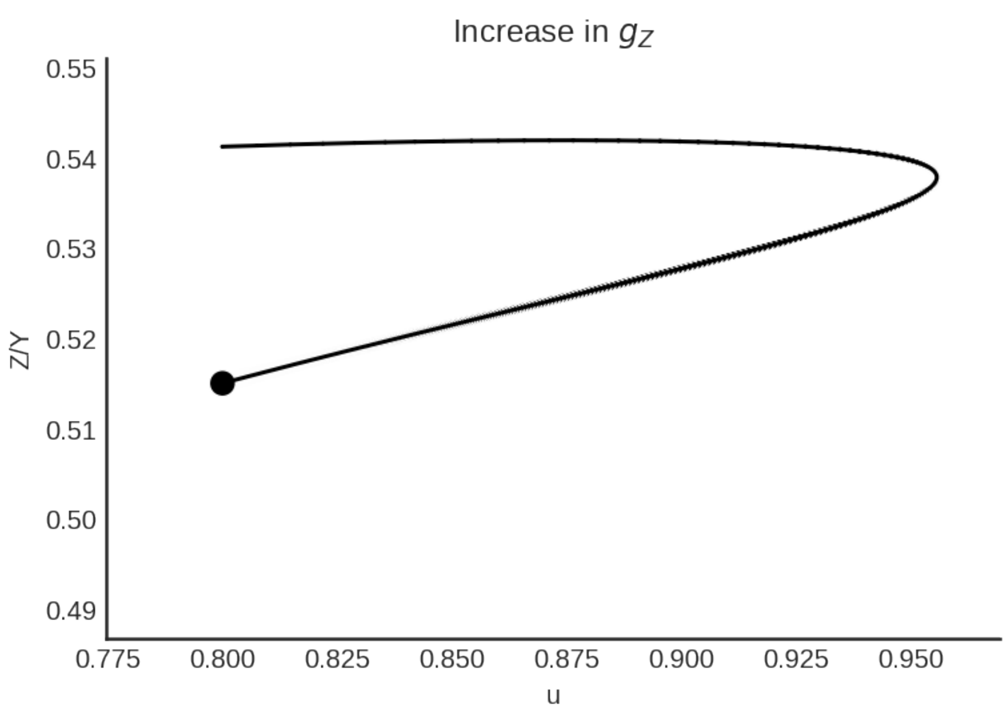
[29]:

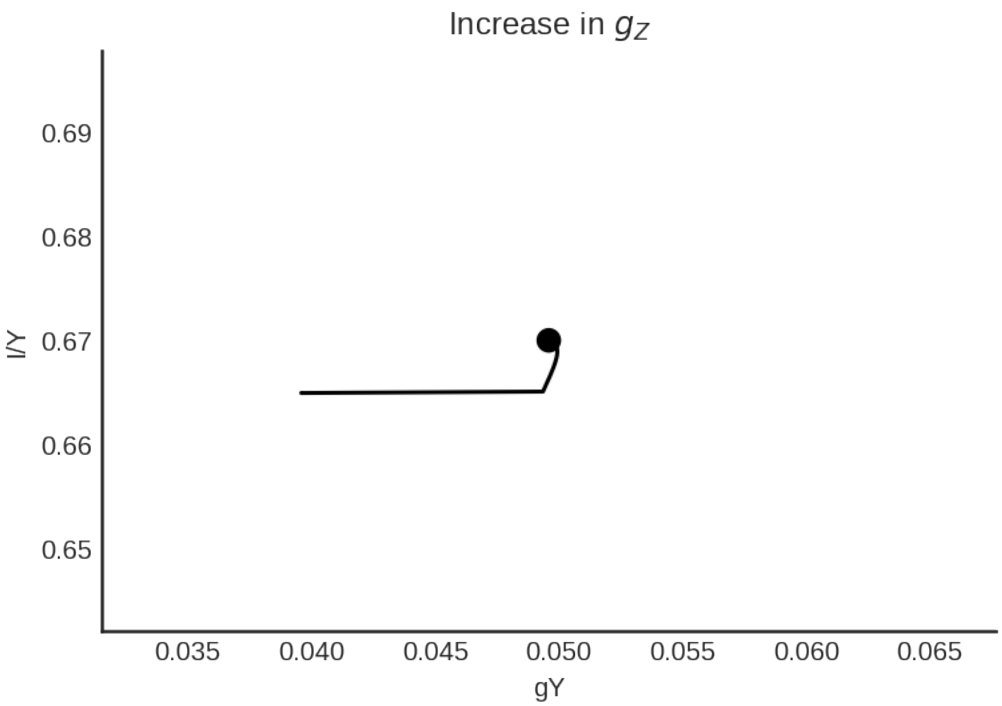
```
           $\Delta \phi_0$
alpha         1.000000e+00
alpha_2       3.000000e-01
gamma_F       9.000000e-01
gamma_u       1.000000e-02
omega         3.000000e-01
rm            2.000000e-02
spread_l      0.000000e+00
spread_mo     0.000000e+00
un            8.000000e-01
v             2.500000e+00
phi_0         5.000000e-02
phi_1         2.000000e-02
phparam       1.000000e+00
infla         0.000000e+00
gZn           3.922000e-02
_K_f__1       1.082934e+44
_M__1        -1.611049e+44
_MO__1        3.598007e+44
_Lf__1       -5.209056e+44
_h__1         1.550000e-01
_L__1        -5.209056e+44
_K_HD__1      3.598007e+44
_I_h__1       1.700278e+43
```

```
_ph__1        1.000000e+00
```

[30]:
```
shock["TIME"] = [i+1 for i in range(len(shock.index))]
shock["Ih/Y"] = shock["I_h"]/shock["Y"]
shock["I/Y"] = shock["I_t"]/shock["Y"]
shock["Z/Y"] = shock["Z"]/shock["Y"]
shock["gY"] = shock["Y"].pct_change()

sns.scatterplot(y = 'Ih/Y', x='u', data=shock, size="TIME", sizes = (1,100),␣
 ↪color = 'black', legend=False)
sns.lineplot(y = 'Ih/Y', x='u', data=shock, sort=False, color = 'black')
plt.title('Increase in $g_Z$')
sns.despine()
plt.show()

sns.scatterplot(y = 'Z/Y', x='u', data=shock, size="TIME", sizes = (1,100),␣
 ↪color = 'black', legend=False)
sns.lineplot(y = 'Z/Y', x='u', data=shock, sort=False, color = 'black')
plt.title('Increase in $g_Z$')
sns.despine()
plt.show()

sns.scatterplot(y = 'I/Y', x='gY', data=shock, size="TIME", sizes = (1,100),␣
 ↪color = 'black', legend=False)
sns.lineplot(y = 'I/Y', x='gY', data=shock, sort=False, color = 'black')
plt.title('Increase in $g_Z$')
sns.despine()
plt.show()
```

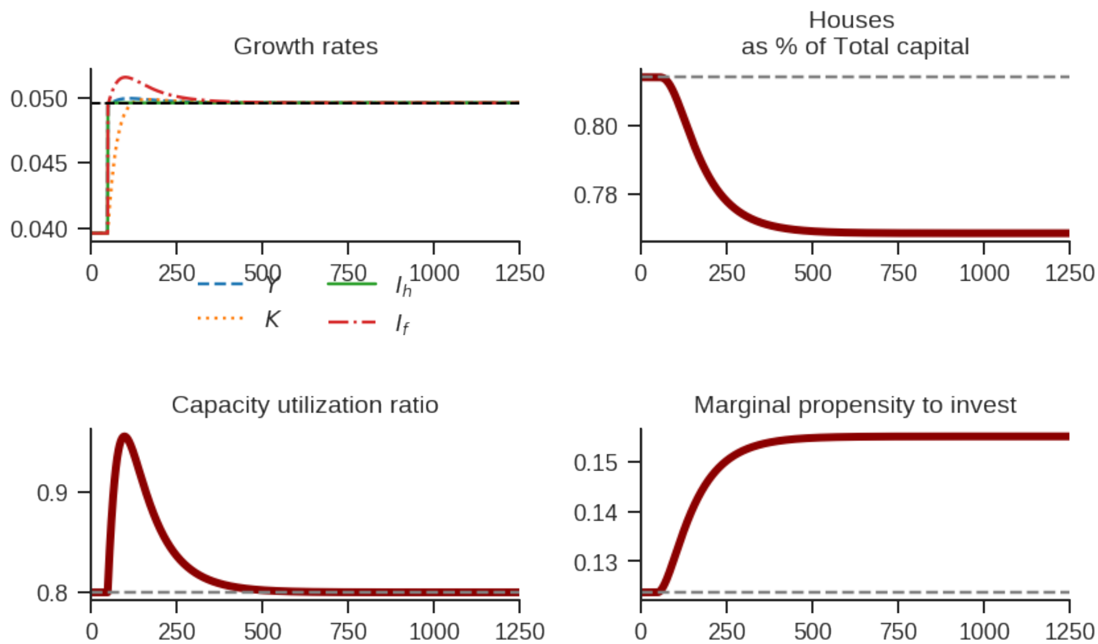

Increase in g_Z
0.55
0.54
0.53
0.52
0.51
0.50
0.49
Ih/Y
0.775
0.800
0.825
0.850
0.875
0.900
0.925
0.950
u


Increase in g_Z
0.55
0.54
0.53
0.52
0.51
0.50
0.49
Z/Y
0.775
0.800
0.825
0.850
0.875
0.900
0.925
0.950
u



9.1.1 Plots

[31]:
```
plot_shock(shock = shock, filename = 'Shock_1.png')
```



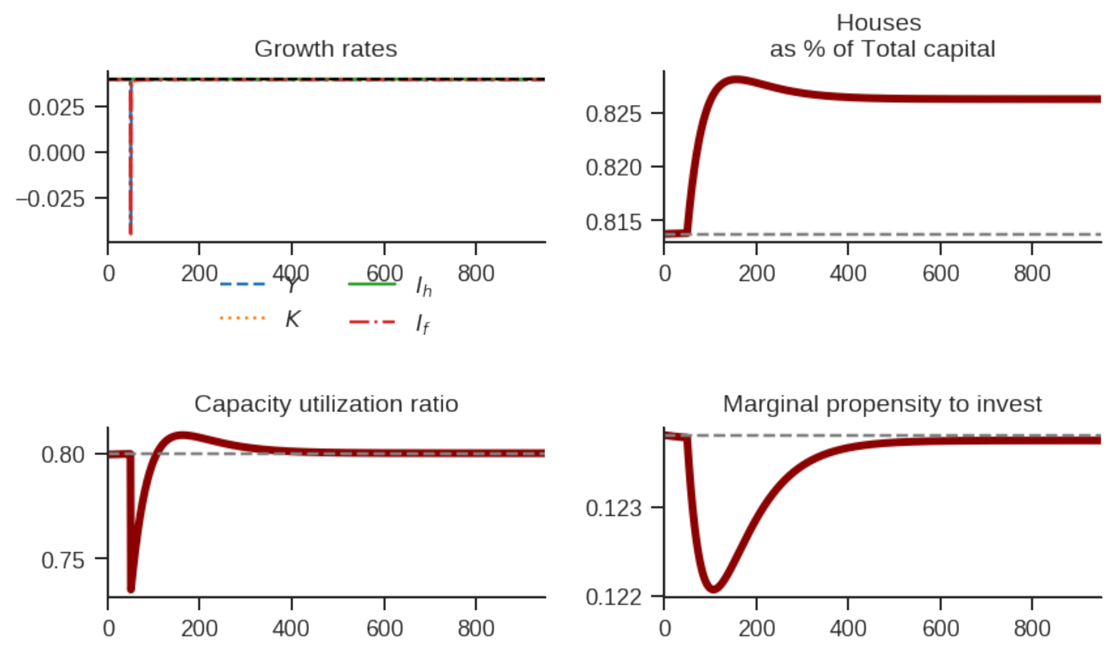
9.2 Decrease in wage-share (ω)

[32]:
```
base = model()
df = SolveSFC(base, time=700)
shock = ShockModel(base_model=base, create_function=model(), variable='omega',␣
 ↪increase=-0.05, time = 900)
shock.transpose()
```

[32]:
```
                    0              1              2              3    \
C          3.964712e+13   4.121709e+13   4.284923e+13   4.454601e+13
C_w        3.549846e+13   3.690415e+13   3.836550e+13   3.988472e+13
C_k        4.148657e+12   4.312940e+12   4.483730e+12   4.661283e+12
FD         1.382886e+13   1.437647e+13   1.494577e+13   1.553761e+13
Fn         1.382886e+14   1.437647e+14   1.494577e+14   1.553761e+14
...                 ...            ...            ...            ...
_h__1      1.238098e-01   1.238098e-01   1.238090e-01   1.238083e-01
_L__1     -2.882750e+15  -2.882750e+15  -2.996908e+15  -3.115587e+15
_K_HD__1   1.680973e+15   1.680973e+15   1.747539e+15   1.816742e+15
_I_h__1    6.403090e+13   6.403090e+13   6.656652e+13   6.920255e+13
_ph__1     1.000000e+00   1.000000e+00   1.000000e+00   1.000000e+00

                    4              5              6              7    \
C          4.630997e+13   4.814379e+13   5.005022e+13   5.203215e+13
C_w        4.146410e+13   4.310603e+13   4.481297e+13   4.658750e+13
C_k        4.845866e+12   5.037759e+12   5.237251e+12   5.444643e+12
FD         1.615289e+13   1.679253e+13   1.745750e+13   1.814881e+13
Fn         1.615289e+14   1.679253e+14   1.745750e+14   1.814881e+14
...                 ...            ...            ...            ...
_h__1      1.238075e-01   1.238068e-01   1.238061e-01   1.238054e-01
_L__1     -3.238965e+15  -3.367229e+15  -3.500573e+15  -3.639197e+15
_K_HD__1   1.888685e+15   1.963477e+15   2.041230e+15   2.122063e+15
_I_h__1    7.194297e+13   7.479192e+13   7.775368e+13   8.083272e+13
_ph__1     1.000000e+00   1.000000e+00   1.000000e+00   1.000000e+00

                    8              9    ...            941            942   \
C          5.409255e+13   5.623455e+13  ...   2.327875e+29   2.420059e+29
C_w        4.843231e+13   5.035016e+13  ...   2.022041e+29   2.102114e+29
C_k        5.660248e+12   5.884390e+12  ...   3.058338e+28   3.179448e+28
FD         1.886749e+13   1.961463e+13  ...   1.019446e+29   1.059816e+29
Fn         1.886749e+14   1.961463e+14  ...   1.019446e+30   1.059816e+30
...                 ...            ...  ...            ...            ...
_h__1      1.238046e-01   1.238040e-01  ...   1.237499e-01   1.237499e-01
_L__1     -3.783310e+15  -3.933131e+15  ...  -2.064168e+31  -2.145909e+31
_K_HD__1   2.206097e+15   2.293458e+15  ...   1.201864e+31   1.249458e+31
```

```
_I_h__1    8.403370e+13  8.736143e+13  ...  4.578088e+29  4.759381e+29
_ph__1     1.000000e+00  1.000000e+00  ...  1.000000e+00  1.000000e+00

                    943           944           945           946  \
C          2.515893e+29  2.615523e+29  2.719097e+29  2.826774e+29
C_w        2.185358e+29  2.271898e+29  2.361865e+29  2.455395e+29
C_k        3.305354e+28  3.436246e+28  3.572321e+28  3.713785e+28
FD         1.101785e+29  1.145415e+29  1.190774e+29  1.237928e+29
Fn         1.101785e+30  1.145415e+30  1.190774e+30  1.237928e+30
...                 ...           ...           ...           ...
_h__1      1.237499e-01  1.237499e-01  1.237499e-01  1.237499e-01
_L__1     -2.230887e+31 -2.319230e+31 -2.411071e+31 -2.506550e+31
_K_HD__1   1.298936e+31  1.350374e+31  1.403849e+31  1.459441e+31
_I_h__1    4.947852e+29  5.143787e+29  5.347481e+29  5.559241e+29
_ph__1     1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00

                    947           948           949           950
C          2.938714e+29  3.055087e+29  3.176068e+29  3.301841e+29
C_w        2.552629e+29  2.653713e+29  2.758800e+29  2.868048e+29
C_k        3.860851e+28  4.013741e+28  4.172685e+28  4.337924e+28
FD         1.286950e+29  1.337914e+29  1.390895e+29  1.445975e+29
Fn         1.286950e+30  1.337914e+30  1.390895e+30  1.445975e+30
...                 ...           ...           ...           ...
_h__1      1.237499e-01  1.237499e-01  1.237499e-01  1.237499e-01
_L__1     -2.605809e+31 -2.708999e+31 -2.816275e+31 -2.927800e+31
_K_HD__1   1.517235e+31  1.577318e+31  1.639779e+31  1.704715e+31
_I_h__1    5.779387e+29  6.008251e+29  6.246178e+29  6.493526e+29
_ph__1     1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00

[67 rows x 951 columns]
```

[33]:
```
plot_shock(shock = shock, filename = 'Shock_2.png')
```



[34]:
```
shock2 = shock.round(decimals = 3).tail(1).transpose().loc['alpha':,:]
shock2.columns = ['$\Delta \omega$']
shock2
```

[34]:
```
           $\Delta \omega$
alpha         1.000000e+00
alpha_2       3.000000e-01
gamma_F       9.000000e-01
gamma_u       1.000000e-02
omega         2.500000e-01
rm            2.000000e-02
spread_l      0.000000e+00
spread_mo     0.000000e+00
un            8.000000e-01
v             2.500000e+00
phi_0         4.000000e-02
phi_1         2.000000e-02
phparam       1.000000e+00
infla         0.000000e+00
gZn           3.900000e-02
_K_f__1       3.585057e+30
_M__1        -1.223085e+31
_MO__1        1.704715e+31
_Lf__1       -2.927800e+31
_h__1         1.240000e-01
_L__1        -2.927800e+31
_K_HD__1      1.704715e+31
```

```
_I_h__1        6.493526e+29
_ph__1         1.000000e+00
```

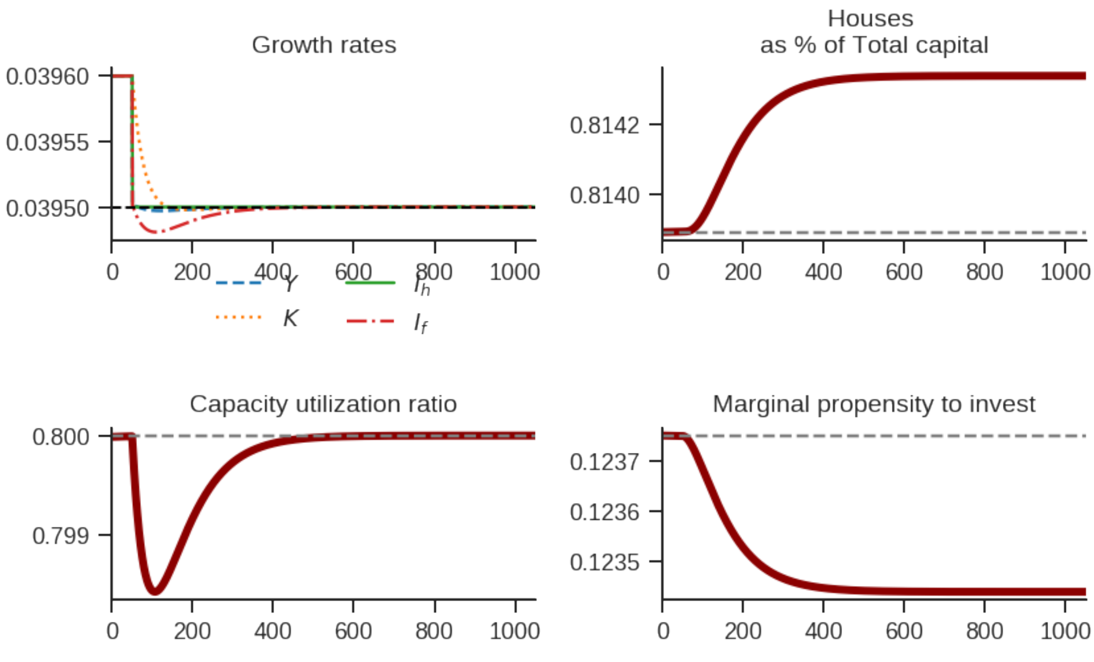
9.3 Increase in interest rates

[35]:
```
base = model()
df = SolveSFC(base, time=1000)
shock = ShockModel(base_model=base, create_function=model(),
 ↪variable='spread_mo', increase=0.005, time = 1000)
shock.transpose()
```

[35]:
```
                    0              1              2              3    \
C           4.550403e+18  4.730599e+18  4.917930e+18  5.112680e+18
C_w         4.074228e+18  4.235567e+18  4.403296e+18  4.577666e+18
C_k         4.761748e+17  4.950313e+17  5.146346e+17  5.350141e+17
FD          1.587249e+18  1.650104e+18  1.715449e+18  1.783380e+18
Fn          1.587249e+19  1.650104e+19  1.715449e+19  1.783380e+19
...                  ...           ...            ...            ...
_h__1       1.237513e-01  1.237513e-01  1.237513e-01  1.237512e-01
_L__1      -3.309027e+20 -3.309027e+20 -3.440065e+20 -3.576292e+20
_K_HD__1    1.929487e+20  1.929487e+20  2.005895e+20  2.085328e+20
_I_h__1     7.349721e+18  7.349721e+18  7.640770e+18  7.943344e+18
_ph__1      1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00

                    4              5              6              7    \
C           5.315142e+18  5.525621e+18  5.744436e+18  5.971915e+18
C_w         4.758941e+18  4.947395e+18  5.143312e+18  5.346987e+18
C_k         5.562007e+17  5.782262e+17  6.011239e+17  6.249284e+17
FD          1.854002e+18  1.927421e+18  2.003746e+18  2.083095e+18
Fn          1.854002e+19  1.927421e+19  2.003746e+19  2.083095e+19
...                  ...           ...            ...            ...
_h__1       1.237512e-01  1.237512e-01  1.237512e-01  1.237512e-01
_L__1      -3.717913e+20 -3.865142e+20 -4.018202e+20 -4.177323e+20
_K_HD__1    2.167907e+20  2.253757e+20  2.343005e+20  2.435788e+20
_I_h__1     8.257900e+18  8.584913e+18  8.924876e+18  9.278301e+18
_ph__1      1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00

                    8              9    ...          1041          1042   \
C           6.208403e+18  6.454256e+18  ...  1.493965e+36  1.552977e+36
C_w         5.558728e+18  5.778853e+18  ...  1.337334e+36  1.390159e+36
C_k         6.496756e+17  6.754027e+17  ...  1.566313e+35  1.628183e+35
FD          2.165585e+18  2.251342e+18  ...  5.221045e+35  5.427276e+35
Fn          2.165585e+19  2.251342e+19  ...  5.221045e+36  5.427276e+36
...                  ...           ...  ...           ...           ...
_h__1       1.237512e-01  1.237511e-01  ...  1.234375e-01  1.234375e-01
_L__1      -4.342745e+20 -4.514717e+20  ... -1.050300e+38 -1.091786e+38
_K_HD__1    2.532246e+20  2.632522e+20  ...  6.110271e+37  6.351627e+37
```

```
_I_h__1   9.645722e+18  1.002769e+19  ...  2.321844e+36  2.413557e+36
_ph__1    1.000000e+00  1.000000e+00  ...  1.000000e+00  1.000000e+00

                  1043          1044          1045          1046  \
C         1.614319e+36  1.678085e+36  1.744369e+36  1.813272e+36
C_w       1.445070e+36  1.502150e+36  1.561485e+36  1.623164e+36
C_k       1.692496e+35  1.759350e+35  1.828844e+35  1.901083e+35
FD        5.641653e+35  5.864499e+35  6.096146e+35  6.336944e+35
Fn        5.641653e+36  5.864499e+36  6.096146e+36  6.336944e+36
...                ...           ...           ...           ...
_h__1     1.234375e-01  1.234375e-01  1.234375e-01  1.234375e-01
_L__1    -1.134912e+38 -1.179741e+38 -1.226341e+38 -1.274781e+38
_K_HD__1  6.602516e+37  6.863316e+37  7.134417e+37  7.416226e+37
_I_h__1   2.508893e+36  2.607994e+36  2.711010e+36  2.818095e+36
_ph__1    1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00

                  1047          1048          1049          1050
C         1.884896e+36  1.959350e+36  2.036744e+36  2.117195e+36
C_w       1.687279e+36  1.753926e+36  1.823206e+36  1.895223e+36
C_k       1.976176e+35  2.054235e+35  2.135377e+35  2.219725e+35
FD        6.587254e+35  6.847450e+35  7.117924e+35  7.399082e+35
Fn        6.587254e+36  6.847450e+36  7.117924e+36  7.399082e+36
...                ...           ...           ...           ...
_h__1     1.234375e-01  1.234375e-01  1.234375e-01  1.234375e-01
_L__1    -1.325135e+38 -1.377478e+38 -1.431888e+38 -1.488448e+38
_K_HD__1  7.709167e+37  8.013679e+37  8.330220e+37  8.659263e+37
_I_h__1   2.929409e+36  3.045121e+36  3.165403e+36  3.290437e+36
_ph__1    1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00

[67 rows x 1051 columns]
```

[36]:
```
plot_shock(shock = shock, filename = 'Shock_3.png')
```



[37]:
```
shock3 = shock.round(decimals = 3).tail(1).transpose().loc['alpha':,:]
shock3.columns = ['$\Delta rm$']
shock3
```

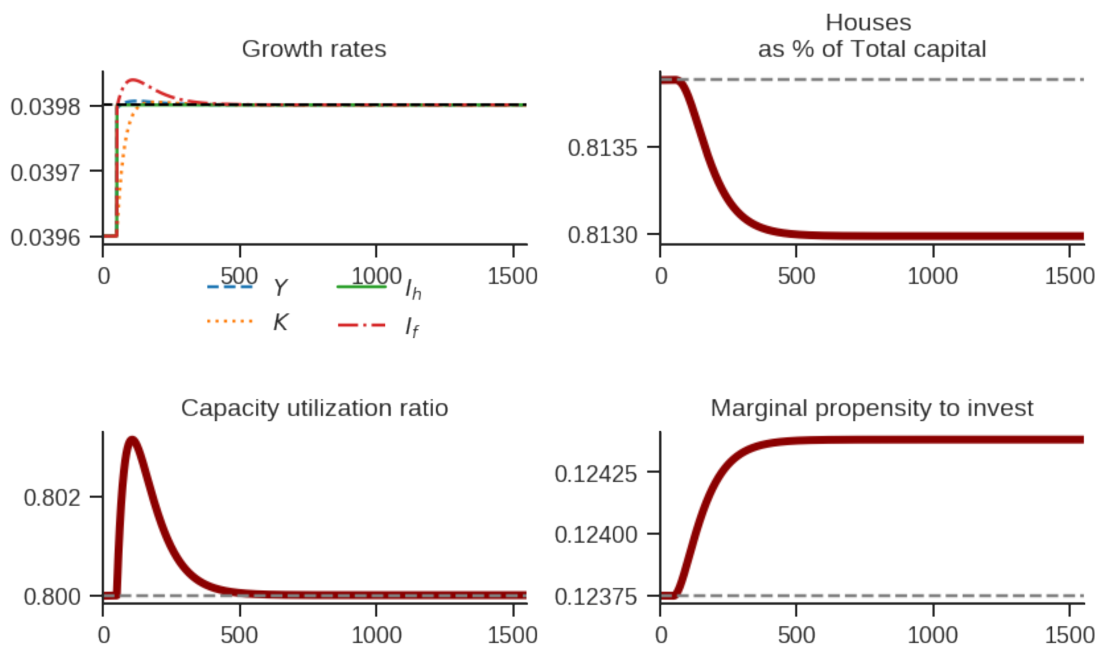
[37]:

	Δrm
alpha	1.000000e+00
alpha_2	3.000000e-01
gamma_F	9.000000e-01
gamma_u	1.000000e-02
omega	3.000000e-01
rm	2.000000e-02
spread_l	0.000000e+00
spread_mo	5.000000e-03
un	8.000000e-01
v	2.500000e+00
phi_0	4.000000e-02
phi_1	2.000000e-02
phparam	1.000000e+00
infla	0.000000e+00
gZn	3.900000e-02
_K_f__1	1.974191e+37
_M__1	-6.225215e+37
_MO__1	8.659263e+37
_Lf__1	-1.488448e+38
_h__1	1.230000e-01
_L__1	-1.488448e+38

```
_K_HD__1    8.659263e+37
_I_h__1     3.290437e+36
_ph__1      1.000000e+00
```

9.4 Increase in inflation

[38]:
```
base = model()
df = SolveSFC(base, time=1000)
shock = ShockModel(base_model=base, create_function=model(), variable='infla',␣
 ↪increase=0.01, time = 1500)

plot_shock(shock = shock, filename = 'Shock_4.png')
```



[39]:
```
shock4 = shock.round(decimals = 3).tail(1).transpose().loc['alpha':,:]
shock4.columns = ['$\Delta $ Infla']
shock4
```

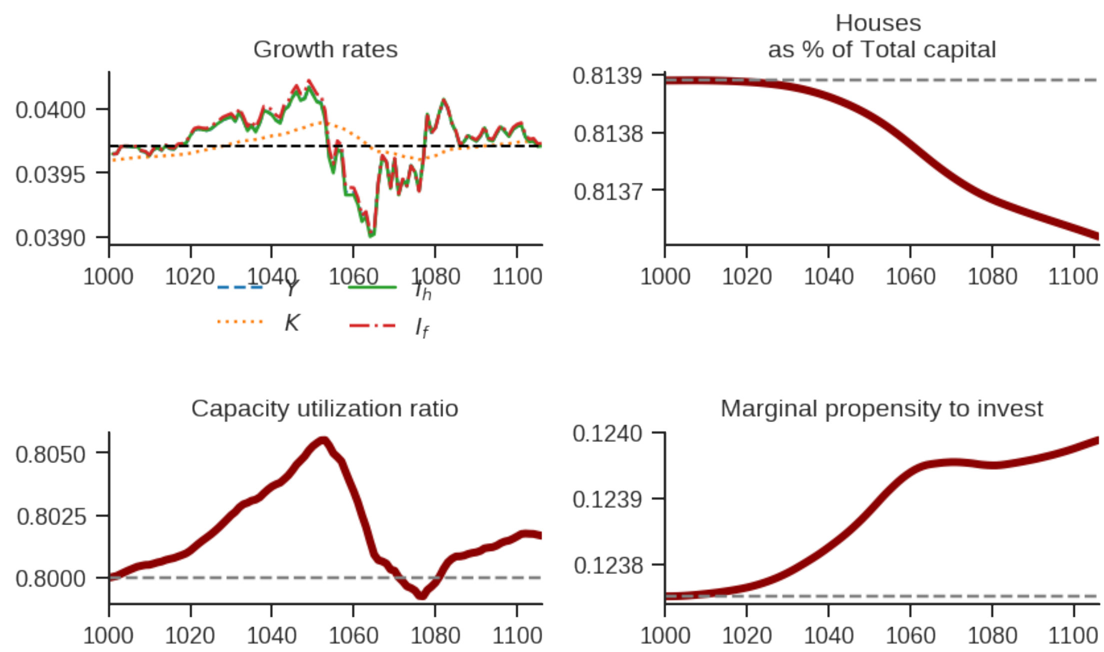
[39]:

	$\Delta $ Infla
alpha	1.000000e+00
alpha_2	3.000000e-01
gamma_F	9.000000e-01
gamma_u	1.000000e-02
omega	3.000000e-01
rm	2.000000e-02
spread_l	0.000000e+00
spread_mo	0.000000e+00

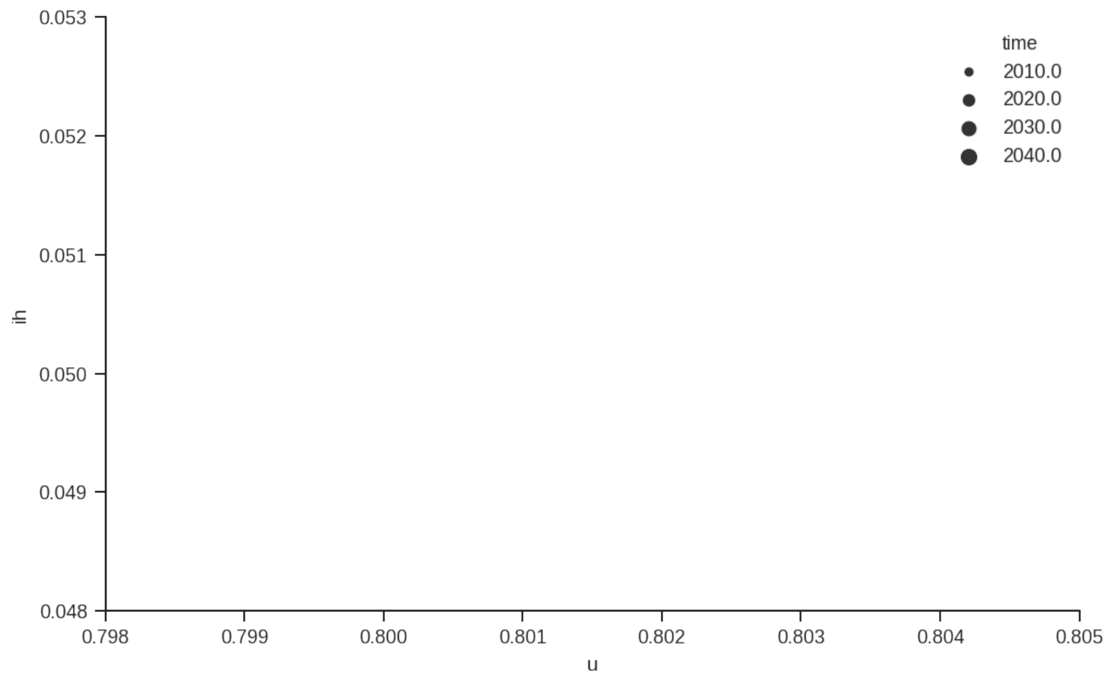
```
un             8.000000e-01
v              2.500000e+00
phi_0          4.000000e-02
phi_1          2.000000e-02
phparam        1.000000e+00
infla          1.000000e-02
gZn            3.900000e-02
_K_f__1        7.896508e+45
_M__1         -2.427467e+46
_MO__1         3.432732e+46
_Lf__1        -5.860199e+46
_h__1          1.240000e-01
_L__1         -5.860199e+46
_K_HD__1       3.432732e+46
_I_h__1        1.313996e+45
_ph__1         3.004272e+06
```

9.5 Plugging real data

[40]:
```python
base=model()

initial=1000
shock_duration=1

SolveSFC(base, time=initial, table=False)

for i in data.index:
    lagged = [key for key in base.solutions[-1].keys()]
    lagged = [i for i in lagged if "__" in i]
    for j in lagged:
        del base.solutions[-1][j]
    base.set_values(base.solutions[-1])
    base.set_values({
            'own':data['Taxa Própria'][i],
            'infla':data['Inflação'][i],
        })
    try:
        SolveSFC(base, time=shock_duration, table=False)
    except Exception as e:
        print(f'For own interest rate = {i}, {e}')
        pass

shock = SFCTable(base)[initial:]
plot_shock(shock = shock, filename = 'Shock_5.png')
```



[41]:
```
shock['ih'] = shock['I_h']/shock['Y']
shock['time'] = (shock.index/4) - shock.shape[0] + (1987 - 120)

cycle=int(shock.shape[0])

for i in range(0,shock.shape[0],cycle):

    fig, ax = plt.subplots(figsize=(8,5))

    sns.scatterplot(x='u', y='ih', data=shock[i:i+cycle], size='time',␣
↪color='black')
    sns.lineplot(x = 'u', y='ih', data=shock[i:i+cycle], ax=ax, color =␣
↪'black', sort=False,)
    ax.set_xlim(.798,.805)
    ax.set_ylim(.048,.053)
    sns.despine()
```



10 Merging tables

[42]:
```
df = df.round(decimals = 3).tail(1).transpose().loc['alpha':,:]
df.columns = ['Base scenario']
df

table = pd.merge(left = df, right = shock1, left_index = True, right_index =␣
 ↪True)
table = pd.merge(left = table, right = shock2, left_index = True, right_index =␣
 ↪True)
table = pd.merge(left = table, right = shock3, left_index = True, right_index =␣
 ↪True)
table = pd.merge(left = table, right = shock4, left_index = True, right_index =␣
 ↪True)
table.to_latex("table.tex", bold_rows = True, column_format = 'c', decimal =␣
 ↪',', escape=False,)
table
```

[42]:
```
             Base scenario   $\Delta \phi_0$   $\Delta \omega$    $\Delta rm$  \
alpha         1.000000e+00      1.000000e+00      1.000000e+00   1.000000e+00
alpha_2       3.000000e-01      3.000000e-01      3.000000e-01   3.000000e-01
gamma_F       9.000000e-01      9.000000e-01      9.000000e-01   9.000000e-01
gamma_u       1.000000e-02      1.000000e-02      1.000000e-02   1.000000e-02
omega         3.000000e-01      3.000000e-01      2.500000e-01   3.000000e-01
```

rm	2.000000e-02	2.000000e-02	2.000000e-02	2.000000e-02
spread_l	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
spread_mo	0.000000e+00	0.000000e+00	0.000000e+00	5.000000e-03
un	8.000000e-01	8.000000e-01	8.000000e-01	8.000000e-01
v	2.500000e+00	2.500000e+00	2.500000e+00	2.500000e+00
phi_0	4.000000e-02	5.000000e-02	4.000000e-02	4.000000e-02
phi_1	2.000000e-02	2.000000e-02	2.000000e-02	2.000000e-02
phparam	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
infla	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
gZn	3.900000e-02	3.922000e-02	3.900000e-02	3.900000e-02
_K_f__1	4.244058e+19	1.082934e+44	3.585057e+30	1.974191e+37
_M__1	-1.326991e+20	-1.611049e+44	-1.223085e+31	-6.225215e+37
_MO__1	1.855990e+20	3.598007e+44	1.704715e+31	8.659263e+37
_Lf__1	-3.182981e+20	-5.209056e+44	-2.927800e+31	-1.488448e+38
_h__1	1.240000e-01	1.550000e-01	1.240000e-01	1.230000e-01
_L__1	-3.182981e+20	-5.209056e+44	-2.927800e+31	-1.488448e+38
_K_HD__1	1.855990e+20	3.598007e+44	1.704715e+31	8.659263e+37
_I_h__1	7.069758e+18	1.700278e+43	6.493526e+29	3.290437e+36
_ph__1	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00

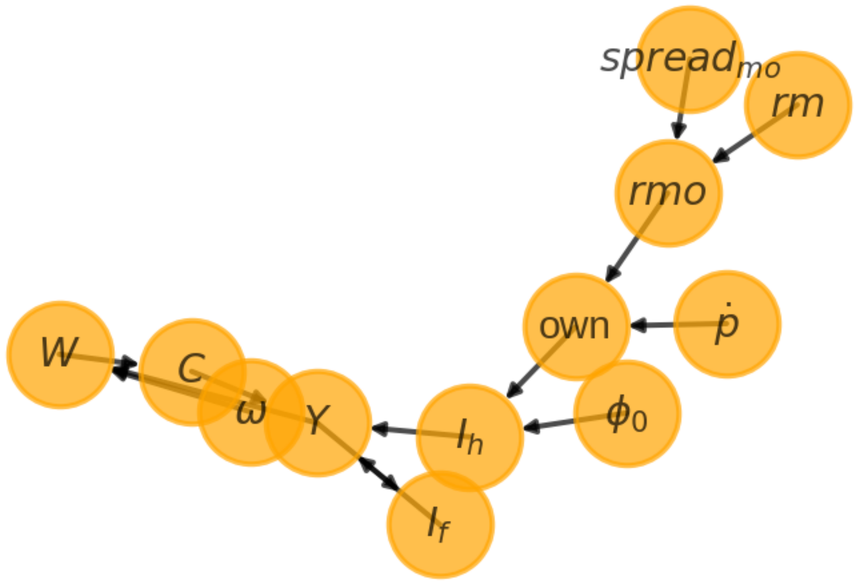
	$\Delta $ Infla
alpha	1.000000e+00
alpha_2	3.000000e-01
gamma_F	9.000000e-01
gamma_u	1.000000e-02
omega	3.000000e-01
rm	2.000000e-02
spread_l	0.000000e+00
spread_mo	0.000000e+00
un	8.000000e-01
v	2.500000e+00
phi_0	4.000000e-02
phi_1	2.000000e-02
phparam	1.000000e+00
infla	1.000000e-02
gZn	3.900000e-02
_K_f__1	7.896508e+45
_M__1	-2.427467e+46
_MO__1	3.432732e+46
_Lf__1	-5.860199e+46
_h__1	1.240000e-01
_L__1	-5.860199e+46
_K_HD__1	3.432732e+46
_I_h__1	1.313996e+45
_ph__1	3.004272e+06

11 Dag

[43]:
```
df = pd.DataFrame(
    {
        'Determinada' : [
            '$Y$', # 1
            #'$Y$', # 2
            '$C$', # 3
            '$W$', # 5
            '$Y$', # 7
            '$Y$', # 8
            '$I_h$', # 9
            'own', # 10
            'own', # 11
            '$I_f$', # 12
            '$I_h$', # 13
            '$rmo$', # 14
            '$rmo$', # 15,
            '$W$', # 16
        ],
        'Determina' : [
            '$C$',  # 1
            #'$It$', # 2
            '$W$', # 3
            '$\omega$', # 5
            '$I_f$', # 7
            '$I_h$', # 8
            'own', # 9
            '$\dot p$', # 10
            '$rmo$', # 11
            '$Y$', # 12,
            '$\phi_0$', # 13
            '$rm$', # 14
            '$spread_{mo}$', # 15
            '$Y$', # 16
        ]
}
)

# Build your graph
G=nx.from_pandas_edgelist(
    df,
    'Determina',
    'Determinada',
    create_using=nx.DiGraph() # To use with arrows
)
```

```python
fig, ax = plt.subplots()

# Plot it
nx.draw(
    G,
    with_labels=True,
    node_color='orange',
    node_size=1500,
    edge_color='black',
    linewidths=2,
    width=2,
    arrows=True,
    font_size=15,
    alpha=.7,
    ax = ax,
    #pos=nx.spring_layout(G),
    pos=nx.fruchterman_reingold_layout(G),
    #pos=nx.shell_layout(G),
)

plt.show()
fig.savefig('Dag.png', dpi = 300)
```

```
/home/gpetrini/.local/lib/python3.6/site-
packages/networkx/drawing/nx_pylab.py:579: MatplotlibDeprecationWarning:
The iterable function was deprecated in Matplotlib 3.1 and will be removed in
3.3. Use np.iterable instead.
  if not cb.iterable(width):
/home/gpetrini/.local/lib/python3.6/site-
packages/networkx/drawing/nx_pylab.py:676: MatplotlibDeprecationWarning:
The iterable function was deprecated in Matplotlib 3.1 and will be removed in
3.3. Use np.iterable instead.
  if cb.iterable(node_size):  # many node sizes
```



12 Analytical solution

[44]:
```
base_eq = model()
SolveSFC(base_eq, time=1, table = False)
t = sp.Symbol('t')
initials = {
    key: base_eq.evaluate(key) for key in base_eq.parameters
}
initials.update({key: base_eq.evaluate(key) for key in base_eq.variables})

for i in base_eq.variables:
  globals()["_" + i] = sp.Function(i)

for i in base_eq.parameters:
  globals()[i] = sp.symbols(i, positive=True)
  globals()['infla'] = sp.symbols('infla')
```

12.1 General equations

[45]:
```
Y = _C(t) + _I_t(t)
pprint(sp.Eq(_Y(t), Y))
I = _I_f(t) + _I_h(t)
pprint(sp.Eq(_I_t(t), I))
Yk = _K_f(t)/v
pprint(sp.Eq(_Yk(t), Yk))
```

```
u = _Y(t)/_Yk(t)
pprint(sp.Eq(_u(t), u))
Z = _I_h(t)
pprint(sp.Eq(_Z(t), Z))
W = omega*_Y(t)
pprint(sp.Eq(_W(t), W))
K = _K_HD(t) + _K_f(t)
pprint(sp.Eq(_K(t), K))
```

```
Y(t) = C(t) + I(t)
I(t) = I_f(t) + I(t)
        K_f(t)
Yk(t) =
          v
        Y(t)
u(t) =
       Yk(t)
Z(t) = I(t)
W(t) = Y(t)
K(t) = K_HD(t) + K_f(t)
```

12.2 Households

[46]:
```
C = alpha*_W(t)
pprint(sp.Eq(_C(t), C))
YD = _W(t) + _FD(t) + rm*_M(t-1) - _rmo(t)*_MO(t-1)
pprint(sp.Eq(_YD(t), YD))
S_h = _YD(t) - _C(t)
pprint(sp.Eq(_S_h(t), S_h))
dMO = _I_h(t)
pprint(sp.Eq(_MO(t) - _MO(t-1), dMO))
V_h = _M(t) + _K_HD(t) - _MO(t)
pprint(sp.Eq(_V_h(t), V_h))
NFW_h = _S_h(t) - _I_h(t)
pprint(sp.Eq(_NFW_h(t), NFW_h))
```

```
C(t) = W(t)


      ␣
↪---------------------------------------------------------------------------

      NameError                                 Traceback (most recent call␣
↪last)

      <ipython-input-46-8afb2da33411> in <module>
        2 pprint(sp.Eq(_C(t), C))
```

```
      3 YD = _W(t) + _FD(t) + rm*_M(t-1) - _rmo(t)*_MO(t-1)
----> 4 pprint(sp.Eq(_YD(t), YD))
      5 S_h = _YD(t) - _C(t)
      6 pprint(sp.Eq(_S_h(t), S_h))


    NameError: name '_YD' is not defined
```

12.3 Firms

[ ]:
```
I_f = _h(t)*_Y(t)
pprint(sp.Eq(_I_f(t), I_f))
dK_f = _I_f(t)
pprint(sp.Eq(_K_f(t) - _K_f(t-1), dK_f))
Lf = _I_f(t) - _FU(t) + _L(t-1)
pprint(sp.Eq(_Lf(t), Lf))
FT = _FU(t) + _FD(t)
pprint(sp.Eq(_FT(t), FT))
FU = gamma_F*(_FT(t) - _rl(t)*_L(t-1))
pprint(sp.Eq(_FU(t), FU))
FD = (1 - gamma_F)*(_FT(t) - _rl(t)*_L(t-1))
pprint(sp.Eq(_FD(t), FD))
h = _h(t-1)*gamma_u*(_u(t)-un) + _h(t-1)
pprint(sp.Eq(_h(t), h))
NFW_f = _FU(t) - _I_f(t)
pprint(sp.Eq(_NFW_f(t), NFW_f))
V_f = _K_f(t) - _L(t)
pprint(sp.Eq(_V_f(t), V_f))
```

12.4 Banks

[ ]:
```
L = _Lf(t)
pprint(sp.Eq(_L(t), L))
M = (_L(t) - _L(t-1)) + (_MO(t) - _MO(t-1)) + _M(t-1)
pprint(sp.Eq(_M(t), M))
rmo = rm + spread_mo
pprint(sp.Eq(_rmo(t), rmo))
rl = rm + spread_l
pprint(sp.Eq(_rl(t), rl))
V_b = _L(t) + _MO(t) - _M(t)
pprint(sp.Eq(_V_b(t), V_b))
NFW_b = _rl(t)*_L(t-1) + _rmo(t)*_MO(t-1) - rm*_M(t-1)
pprint(sp.Eq(_NFW_b(t), NFW_b))
```

12.5 Residential Investment

[ ]:
```
_own = sp.Function('own')

K_HS = _K_HD(t)
pprint(sp.Eq(_K_HS(t), K_HS))
Is = _I_h(t)
pprint(sp.Eq(_Is(t), Is))
dK_HD = _I_h(t)
pprint(sp.Eq(_K_HD(t) - _K_HD(t-1), dK_HD))
I_h = (1+_g_Z(t))*_I_h(t-1)
pprint(sp.Eq(_I_h(t), I_h))
K_k = _K_HD(t)/(_K(t))
pprint(sp.Eq(_K_k(t), K_k))
ph = (1+infla)*_ph(t-1)
pprint(sp.Eq(_ph(t), ph))
own = ((1+_rmo(t))/(1+infla))-1
pprint(sp.Eq(_own(t), own))
g_Z = phi_0 - phi_1*_own(t)
pprint(sp.Eq(_g_Z(t), g_Z))
```

12.5.1 Stability condition

[ ]:
```
g = sp.Function('g')
gK = sp.Function('g_K')


def replacer(express):
    #print("\nReplacing the initial values.....")
    df = SolveSFC(model(), time=1)
    df = df.iloc[1, :]

    express = express.subs(alpha, df['alpha']).subs(
        omega, df['omega'])
    express = express.subs(un, df['un']).subs(
        gamma_u, df['gamma_u'])
    express = express.subs(
        infla, df['infla'])
    express = express.subs(phi_0, df['phi_0']).subs(
        phi_1,
        df['phi_1']).subs(rm, df['rm']).subs(
            spread_mo, df['spread_mo'])
    express = express.subs(rm, df['rm']).subs(
            spread_mo, df['spread_mo']).subs(v, df['v'])
    return express
```

[ ]:
```
EqY = Y - _Y(t)
EqY = EqY.subs(_C(t), C).subs(_I_t(t), I)
EqY = EqY.subs(_I_f(t), I_f)
EqY = EqY.subs(_W(t), W)
EqY = EqY.subs(_I_h(t), _Z(t))

EqY = sp.solve(EqY, _Y(t))[0].collect(alpha).collect(omega)
solY = EqY
pprint(cse(solY, optimizations='basic')[1], use_unicode=True)
print('dY/d alpha = ', EqY.diff(alpha))
print('dY/d omega = ', EqY.diff(omega))

print("\nGowth rate.....")
gY, h_, gz_ = sp.symbols('gY h gZ')
gY_ = alpha * omega * gY + _h(t) * gY + _h(t) - _h(
    t - 1) + (_Z(t) / _Y(t)) * _g_Z(t) - gY
gY_ = gY_.subs(_g_Z(t), gz_).subs(_Y(t), solY).subs(_g_Z(t), gz_)
gY_ = gY_.subs(_h(t) - _h(t - 1), h - _h(t - 1))
gY_ = sp.solve(gY_, gY)[0].collect(gz_)
pprint(sp.Eq(g(t), sp.simplify(gY_)))

print('\nd gY/ d alpha\n')
pprint(gY_.diff(alpha))
print('\nd gY/ d omega\n')
pprint(gY_.diff(omega))
```

12.5.2 Stability conditon (I)

[ ]:
```
own_ = sp.Symbol('own')
g_LR = gY_.subs(_u(t), un)
pprint(sp.Eq(g(t), g_LR))

Equ = _u(t)*(g(t) - gK(t)) + _u(t-1)
pprint(sp.Eq(_u(t), Equ))
g_K = (_h(t)*_u(t))/v
pprint(sp.Eq(gK(t), g_K))
Equ = _u(t)*(g(t) - g_K)
Equ = Equ.subs(g(t), gz_)
pprint(sp.Eq(_u(t), Equ))
print(sp.latex(sp.Eq(_u(t), Equ)))

Eqh = _h(t)*gamma_u*(_u(t) - un)
pprint(sp.Eq(_h(t), Eqh))
print(sp.latex(sp.Eq(_h(t), Eqh)))

print('\nBuilding Jacobian matrix and evaluating at u = un\n')
```

```
J = sp.Matrix([
        [
            Eqh.diff(_h(t)).subs(_u(t), un).subs(_h(t-1), _h(t)).subs(_h(t),␣
 ↪g_LR*v/un),
            Eqh.diff(_u(t)).subs(_u(t), un).subs(_h(t-1), _h(t)).subs(_h(t),␣
 ↪g_LR*v/un)
        ],
        [
            Equ.diff(_h(t)).subs(_u(t), un).subs(_h(t-1), _h(t)).subs(_h(t),␣
 ↪g_LR*v/un),
            Equ.diff(_u(t)).subs(_u(t), un).subs(_h(t-1), _h(t)).subs(_h(t),␣
 ↪g_LR*v/un)
        ]])
pprint(J)
print(sp.latex(J))
print('\nDeterminant:\n')
pprint(J.det()>0)
print(sp.latex(J.det()>0))
estability = sp.solve(J.det().subs(gz_, g_Z).subs(_own(t), own_)>0, own_)
print("\nStability condition\n")
pprint(estability)
estability = replacer(estability)
pprint(estability)
pprint(sp.Eq(own_, initials['own']))
estability = estability.subs(own_, own)
pprint(estability)
print('Rewriting')
estability = -(1+infla) + (1+rmo) < 2*(1+infla)
pprint(estability)
estability = sp.solve(estability, infla)
pprint(estability)
estability = estability.subs(spread_mo, initials['spread_mo']).subs(rm,␣
 ↪initials['rm'])
pprint(estability)

print('\nTrace:\n')
pprint(J.trace()<0)
print(sp.latex(J.trace()<0))
print("\nStability condition\n")
estability = sp.solve(J.trace()<0).subs(gz_, g_Z)
pprint(estability)
estability = estability.subs(_own(t), own).subs(_rmo(t), rmo)
pprint(estability)
estability = estability.subs(spread_mo, initials['spread_mo']).subs(rm,␣
 ↪initials['rm'])
```

```
estability = estability.subs(phi_0, initials['phi_0']).subs(phi_1,␣
↪initials['phi_1'])
pprint(estability)
```

12.5.3 Stability condition (II)

```
[ ]: print('d gY/d u < d gK/ d u')
print('d gY/ d u :')
pprint(gY_.diff(_u(t)).subs(_u(t), un).subs(_h(t-1), _h(t)).subs(_h(t), gz_*v/
↪un))

print('d gK/ d u :')
pprint(g_K.diff(_u(t)).subs(_u(t), un).subs(_h(t-1), _h(t)).subs(_h(t), gz_*v/
↪un))

print('Solving the inequality')
pprint(sp.solve(gY_.diff(_u(t)).subs(_u(t), un).subs(_h(t-1), _h(t)).
↪subs(_h(t), gz_*v/un)< g_K.diff(_u(t)).subs(_u(t), un).subs(_h(t-1), _h(t)).
↪subs(_h(t), gz_*v/un), gamma_u))
print(sp.latex(sp.solve(gY_.diff(_u(t)).subs(_u(t), un).subs(_h(t-1), _h(t)).
↪subs(_h(t), gz_*v/un)< g_K.diff(_u(t)).subs(_u(t), un).subs(_h(t-1), _h(t)).
↪subs(_h(t), gz_*v/un), gamma_u)))
```

12.6 Capacity utilization on the long-run

Consider k as the fraction between real housing and total capital (including households' capital):

$$k = \frac{K_h}{K}$$

The capacity utilization ration can be definede as:

$$u = \frac{Y \cdot v}{K \cdot (1 - k)}$$

So, dividing Y by houseolds' capital is the same as:

$$\frac{Y}{k \cdot K}$$

Multiplying by v:

$$\frac{Y}{k \cdot K} \cdot v = \frac{Y \cdot v}{K} \cdot \left(\frac{1}{k}\right)$$

Multiplying and dividing by $1 - k$:

$$\frac{Y \cdot v}{K \cdot (1 - k)} \cdot \left(\frac{1 - k}{k}\right) = u \cdot \left(\frac{1 - k}{k}\right)$$

Therefore,

$$Y\frac{v}{K_h} = u \cdot \left(\frac{1-k}{k}\right)$$

$$u = Y\frac{v}{K_h} \cdot \left(\frac{k}{1-k}\right)$$

[ ]:
```
k = sp.Symbol('K_k')

rel = solY*(v/_K_HD(t))*(k/(1-k))
rel = rel.subs(_Z(t), Z)
rel = rel.subs(_I_h(t)/_K_HD(t), g_Z)
rel = rel.subs(_h(t), h)

pprint(rel)
print('\nFor the long run...\n')

rel = rel.subs(_u(t), un).subs(_h(t-1), _h(t))
rel = rel.subs(_h(t), g_Z*v/un)
rel = rel.subs(_own(t), own)
rel = rel.subs(_rmo(t), rmo)

pprint(sp.Eq(_u(t), rel))
print(sp.latex(sp.Eq(_u(t), rel)))

def collector(express):
    express = express.simplify().collect(phi_0).collect(phi_1).collect(v)
    express = express.collect(omega).collect(alpha).collect(un).collect(infla+1)
    express = express.simplify()
    return express


rel = rel - un
rel = sp.solve(rel,k/(1-k))[0]
pprint(sp.Eq(k/(1-k), collector(rel)))
print(sp.latex(sp.Eq(k/(1-k), collector(rel))))
pprint(sp.Eq(k/(1-k), replacer(rel)))
print(f"Error = {replacer(rel) - base.evaluate('K_k/(1-K_k)')}")

print('\n' + '='*70)
print('\nChange in income distribution:\n')
result = collector(rel.diff(omega))
pprint(result)
print('\nReplacing ...\n')
result = replacer(result)
print(result)
print(f'd (k/1-k)/ d omega > 0? {result > 0}')
```

```
print('\n' + '='*70)
print('\nChange in spread:\n')
result = collector(rel.diff(spread_mo))
pprint(result)
print('\nReplacing ...\n')
result = replacer(result)
print(result)
print(f'd (k/1-k)/ d spread > 0? {result > 0}')

print('\n' + '='*70)
print('\nChange in inflation:\n')
result = collector(rel.diff(infla))
pprint(result)
print('\nReplacing ...\n')
result = replacer(result)
print(result)
print(f'd (k/1-k)/ d inflation > 0? {result > 0}')

print('\n' + '='*70)
print('\nChange in autonomous component:\n')
result = collector(rel.diff(phi_0))
pprint(result)
print('\nReplacing ...\n')
result = replacer(result)
print(result)
print(f'd (k/1-k)/ d phi_0 > 0? {result > 0}')

print('\n' + '='*70)
print('\nHousing as % of total Capital\n')
rel = rel*(1-k) - k
rel = sp.solve(rel, k)[0]
pprint(sp.Eq(k, collector(rel)))
print(sp.latex(sp.Eq(k, collector(rel))))
print(f"Error = {replacer(rel) - base.evaluate('K_k')}")
print(f"Error = {replacer(1-rel) - base.evaluate('(g_Z*v/un)/
 ↪(1-alpha*omega)')}")

print('\n' + '='*70)
print('\nChange in income distribution:\n')
result = collector(rel.diff(omega))
pprint(result)
print('\nReplacing ...\n')
result = replacer(result)
print(result)
print(f'd k/ d omega > 0? {result > 0}')

print('\n' + '='*70)
```

```python
print('\nChange in spread:\n')
result = collector(rel.diff(spread_mo))
pprint(result)
print('\nReplacing ...\n')
result = replacer(result)
print(result)
print(f'd k/ d spread > 0? {result > 0}')

print('\n' + '='*70)
print('\nChange in inflation:\n')
result = collector(rel.diff(infla))
pprint(result)
print('\nReplacing ...\n')
result = replacer(result)
print(result)
print(f'd k/ d inflation > 0? {result > 0}')

print('\n' + '='*70)
print('\nChange in autonomous component:\n')
result = collector(rel.diff(phi_0))
pprint(result)
print('\nReplacing ...\n')
result = replacer(result)
print(result)
print(f'd k/ d phi_0 > 0? {result > 0}')

print('\n' + '='*70)
print('\nFirms capital as % of total Capital:\n')
rel = (g_Z*v/un)/(1-omega*alpha)
rel = rel.subs(_own(t), own).subs(_rmo(t), rmo)
pprint(sp.Eq(1-k, rel))

print('\n' + '='*70)
print('\nChange in income distribution:\n')
result = collector(rel.diff(omega))
pprint(result)
print('\nReplacing ...\n')
result = replacer(result)
print(result)
print(f'd k/ d omega > 0? {result > 0}')

print('\n' + '='*70)
print('\nChange in spread:\n')
result = collector(rel.diff(spread_mo))
pprint(result)
print('\nReplacing ...\n')
result = replacer(result)
```

```
print(result)
print(f'd k/ d spread > 0? {result > 0}')

print('\n' + '='*70)
print('\nChange in inflation:\n')
result = collector(rel.diff(infla))
pprint(result)
print('\nReplacing ...\n')
result = replacer(result)
print(result)
print(f'd k/ d inflation > 0? {result > 0}')

print('\n' + '='*70)
print('\nChange in autonomous component:\n')
result = collector(rel.diff(phi_0))
pprint(result)
print('\nReplacing ...\n')
result = replacer(result)
print(result)
print(f'd k/ d phi_0 > 0? {result > 0}')

print('\n' + '='*70)
print('\nRevisiting Houses as % of total Capital:\n')
rel = 1 - rel
pprint(sp.Eq(k, rel))
print(sp.latex(sp.Eq(k, rel)))

print('\n' + '='*70)
print('\nChange in income distribution:\n')
result = collector(rel.diff(omega))
pprint(result)
print(sp.latex(result))
print('\nReplacing ...\n')
result = replacer(result)
print(result)
print(f'd k/ d omega > 0? {result > 0}')

print('\n' + '='*70)
print('\nChange in spread:\n')
result = collector(rel.diff(spread_mo))
pprint(result)
print(sp.latex(result))
print('\nReplacing ...\n')
result = replacer(result)
print(result)
print(f'd k/ d spread > 0? {result > 0}')
```

```
print('\n' + '='*70)
print('\nChange in inflation:\n')
result = collector(rel.diff(infla))
pprint(result)
print(sp.latex(result))
print('\nReplacing ...\n')
result = replacer(result)
print(result)
print(f'd k/ d inflation > 0? {result > 0}')

print('\n' + '='*70)
print('\nChange in autonomous component:\n')
result = collector(rel.diff(phi_0))
pprint(result)
print(sp.latex(result))
print('\nReplacing ...\n')
result = replacer(result)
print(result)
print(f'd k/ d phi_0 > 0? {result > 0}')
```

13 Steady State

[ ]:
```
#pprint(sp.Eq(Y / _K_f(t), I_f / K_HS))
#pprint(sp.Eq(K_HS / _K_f(t), I_f / Y))
#pprint(sp.Eq(rel, I_f / Y))
#flow = I_f / Y
#flow = flow.subs(_Y(t), Y).subs(_h(t), (g_Z.subs(_own(t), own).subs(_rmo(t),␣
 ↪rmo))*v/un)
#pprint(sp.Eq(rel, flow))
#ss = sp.solve(sp.Eq(rel, flow), omega)[0]
#pprint(sp.Eq(omega, ss))
#pprint(sp.Eq(omega, replacer(ss)))
```

14 Sobol

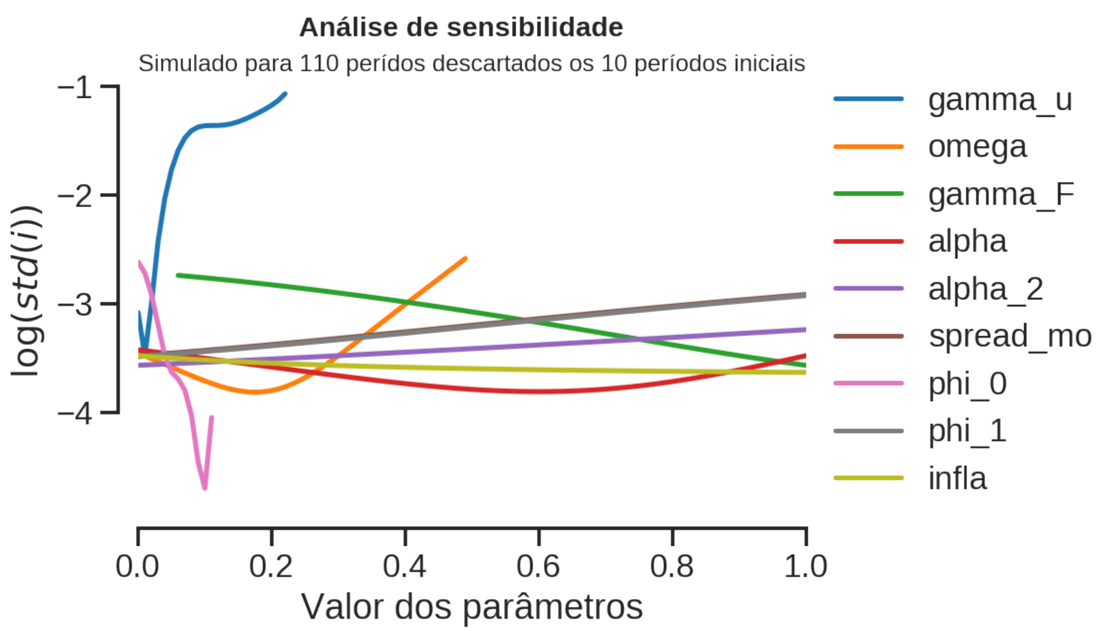
[ ]:
```
t2 = datetime.now()
```

[47]:
```
parameters = [
    'gamma_u',
    'omega',
    'gamma_F',
    'alpha',
    'alpha_2',
    'spread_mo',
    'phi_0',
```

```
    'phi_1',
    'infla',
]

def sobol(
    bound = np.linspace(0,1,101),
    time = 10,
    skip = 10,
    parameters = parameters
):
    t2 = datetime.now()
    bound = bound
    df = pd.DataFrame()
    empty_list = [i for i in range(len(bound))]

    for param in parameters:
        for i in range(len(bound)):
            base = model()
            base.set_values({param:bound[i]})
            try:
                empty_list[i] = np.log(SolveSFC(base,time=time+skip)["u"][skip:
 ↪].std())
            except Exception as e:
                empty_list[i] = np.infty
                pass
        df[param] = empty_list ################## Replace here
################################ End␣
 ↪#####################################################
    df.index = bound

    sns.set_context('talk')
    fig, ax = plt.subplots()

    df.plot(
        ax = ax,
        lw = 2.5
    )
    ax.ticklabel_format(useOffset=False)
    ax.set_ylabel("$\log(std(i))$")
    ax.set_xlabel("Valor dos parâmetros")
    ax.legend(loc='center left', bbox_to_anchor=(1, 0.5))

    fig.suptitle("Análise de sensibilidade", fontsize = 14, weight="bold")
    ax.set_title("Simulado para {} perídos descartados os {} períodos iniciais".
 ↪format(time+skip,skip), fontsize = 12, y = .98)

    sns.despine(offset=10, trim=True);
```

```
    ylim = ax.get_ylim()

    plt.show()
    print("Total running time: ", datetime.now()-t2)
    return df
```

[48]:
```
sobol(time=100)
```




```
Total running time:  0:11:40.250208
```

[48]:
```
       gamma_u     omega   gamma_F     alpha   alpha_2  spread_mo     phi_0  \
0.00 -3.081586 -3.453268       inf -3.422897 -3.565038  -3.476689 -2.617742
0.01 -3.476689 -3.478576       inf -3.430510 -3.562300  -3.472092 -2.716890
0.02 -2.996520 -3.504268       inf -3.438168 -3.559546  -3.467438 -2.915440
0.03 -2.420178 -3.530262       inf -3.445868 -3.556776  -3.462728 -3.198037
0.04 -2.031450 -3.556476       inf -3.453611 -3.553990  -3.457961 -3.476689
...        ...       ...       ...       ...       ...        ...       ...
0.96       inf       inf -3.531529 -3.531065 -3.252454  -2.935542       inf
0.97       inf       inf -3.540162 -3.517688 -3.248877  -2.930463       inf
0.98       inf       inf -3.548632 -3.504162 -3.245296  -2.925420       inf
0.99       inf       inf -3.556927 -3.490487 -3.241711  -2.920412       inf
1.00       inf       inf -3.565038 -3.476689 -3.238122  -2.915440       inf

         phi_1     infla
0.00 -3.485707 -3.476689
```

```
0.01 -3.481228 -3.481272
0.02 -3.476689 -3.485708
0.03 -3.472092 -3.490000
0.04 -3.467438 -3.494154
...        ...       ...
0.96 -2.945803 -3.628871
0.97 -2.940656 -3.629317
0.98 -2.935542 -3.629757
0.99 -2.930463 -3.630190
1.00 -2.925420 -3.630616

[101 rows x 9 columns]
```

15 Finishing

[ ]:
```
print("Total running time: ", datetime.now() - t1)
print("Simulation running time: ", t2 - t1)
print("Sobol running time: ", datetime.now() - t2)
```

16 Shock yourself

[ ]:
```
base = model()
df_ = SolveSFC(base, time=1000)

@interact(variable = (df_.transpose().loc['alpha':,:].index), increase =
 ↪(-1,1,0.01))
def shock_yourself(variable, increase):
  shock = ShockModel(base_model = base, create_function = model(), variable =
 ↪variable, increase = increase)
  plot_shock(shock = shock, filename = 'SelfShock.png')
```