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...

Variables list

- C Consumption
- CG: Capital gains
- FT: Total profits
- FD: Distributed profits
- FU: Retained profits
- g_i: Growth rate of variable i
- g_i^e: Expected growth rate of variable i
- h: Marginal propensity to invest
- I: Total investment
- I_f: Non-residential investment
- I_h: Residential investment
- K_f: Non-residential fixed capital
- K_h: Residential capital
- K_{HD}: Demand for houses
- K_{HU}: Unsold houses
- K_H: Houses supply
- L: Loans
- $morp$: Mortgages repayments
- M: Money deposits
- MO: Mortgages
- p_h Housing prices
- r_l: Interest rates on Loans
- r_m: Interest rates on money deposits
- r_H: Housing own interest rates
- S_i: Sector i savings
- u: Capacity utilization ratio
- v: Capacity-Output ratio
- V_i: Net financial Wealth of sector i
- W Total Wage bill
- Y: GDP
- Y_K: Capacity
- YD: Household disposable income
- Z: Autonomous expenditures

Parameters list

- α: Propensity to consume out of wages
- β: Expectation adjustment parameter
- γ_F: % of distributed profits
- γ_u: Adjustment parameter for the marginal propensity to invest

Exogenous variables

- gz: Autonomous grouth rate
- ω: Wage share
- rm: Interest rates on money deposits
- $spread_l$: Spread for loans
- $spread_{mo}$: Spread for mortgages
- un: Normal capacity utilization ratio
- v: Capitl-Output ratio

Assumptions

General

- No inflation
- Two kinds of capital: Residential and non-Residential
- No depeciation
- Non-residential investment is induced

Households

- All savings are accumulated in bank deposits
- Households do not have access to Loans
- Residential investment is financed by mortgages

Firms

Banks

- Have any net worth

Matrix

Balance sheet

	Households	Firms	Banks	$\sum$
Money Deposits	$+M$		$-M$	0
Loans		$-L$	$+L$	0
Mortgages	$-MO$		$+MO$	0
Capital		K_f		K_f
Houses	K_{HD}			K_H
Net Worth	V_h	V_f	V_b	K

Transactions and flow of funds matrix

	Households	Households	Firms	Firms	Banks	Banks	Sum
	Current	Capital	Current	Capital	Current	Capital	$\sum$
Consumption	$-C$		$+C$				0
Investment		$-I_h$	I	$-I_f$			0
[Production]			$[Y]$				Y
Wages	$+W$		$-W$				0
Profits	$+FD$		$-FT$	$+FU$			0
Interests on loans			$-r_{l-1} \cdot L_{-1}$		$+r_{l-1} \cdot L_{-1}$		0
Interests on Bank deposits	$+r_{m-1} \cdot M_{-1}$				$-r_{m-1} \cdot M_{-1}$		0
Interests on Mortgage	$-r_{mo-1} \cdot MO_{-1}$				$+r_{mo-1} \cdot MO_{-1}$		0
Subtotal	S_h			S_f	S_b		0
Change in Loans				$+\Delta L$		$-\Delta L$	0
Change in Bank deposits	$-\Delta M$					$+\Delta M$	0
Change in Mortgages		$+\Delta MO$				$-\Delta MO$	0
Sum	0	0	0	0	0	0	0

Equations

General Equations

$$Y = C + I \tag{1}$$

$$I = I_f + I_h \tag{2}$$

$$\omega = \overline{\omega} \tag{3}$$

$$W = \omega \cdot Y \tag{4}$$

$$Y_K = \frac{K_f}{\overline{v}} \tag{5}$$

$$u = \frac{Y}{Y_K} \tag{6}$$

$$g_k = \frac{h \cdot u}{v} \tag{7}$$

$$Z = I_h \tag{8}$$

$$K = K_f + K_{HD} \tag{9}$$

Households

$$YD = W + FD + \bar{r}_{m-1} \cdot M_{-1} - \bar{r}_{mo-1} \cdot MO \tag{10}$$

$$S_h = YD - C = \Delta M \tag{11}$$

$$\Delta MO = \Delta M - S_h \tag{12}$$

$$C = \alpha \cdot YD \tag{13}$$

$$V_h = M + K_H - MO \tag{14}$$

Firms

$$\Delta L = I_f - FU \tag{15}$$

$$FT = Y - W = FU + FD \tag{16}$$

$$FU = \gamma_F \cdot (FT - r_{L_{-1}} \cdot L_{-1}) \tag{17}$$

$$FD = (1 - \gamma_F) \cdot (FT - r_{L_{-1}} \cdot L_{-1}) \tag{18}$$

$$I_f = h \cdot Y \tag{19}$$

$$\Delta K_f = I_f \tag{20}$$

$$\Delta h = h_{-1} \cdot \gamma_u \cdot (u - \bar{u}_n) \tag{21}$$

$$V_f = K_f - L \tag{22}$$

$$S_f = FU - I_f \tag{23}$$

Banks

$$\Delta M = \Delta L + \Delta MO \tag{24}$$

$$S_b = rl_{-1} \cdot L_{-1} + rmo_{-1} \cdot MO_{-1} - rm_{-1} \cdot M_{-1} \tag{25}$$

$$r_l = r_m + spread_l \tag{26}$$

$$r_{mo} = r_m + spread_{mo} \tag{27}$$

$$V_b = L + MO - M \tag{28}$$

Residential Investment

$$K_{HS} = K_{HD} \tag{29}$$

$$\Delta K_{HD} = I_h \tag{30}$$

$$I_h = \Delta MO \tag{31}$$

$$g_{I_h} = \bar{g}_{I_h} \tag{32}$$

Simulation setup

Installing required packages

```
!pip install pysolve3
```

Loading libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('seaborn-white')
import seaborn as sns
from pysolve3.utils import ShockModel, SFCTable, SolveSFC
from pysolve3.model import Model
import sympy as sp
%config InlineBackend.figure_format = 'retina'
```

Creating model function

```
def model():
  model = Model()
  model.set_var_default(0)

  model.var('C', desc='Consumption')
  model.var('FD', desc='Distributed profits')
  model.var('FT', desc='Total Profits')
  model.var('FU', desc='Retained profits', default = 100)
  model.var('gk', desc='Capital growth rate', default=0.01)
  model.var('g_Ih', desc='Residential investment growth rate (Demand)', defa
  model.var('h', desc='Marginal propensity to invest (non-residential)', def
  model.var('I_t', desc='Investment')
  model.var('I_f', desc='Non-residential investment', default = 100)
  model.var('I_h', desc='Residential investment', default = 300)
  model.var('K_HS', desc='Houses supply', default=500)
  model.var('K_HD', desc='Houses demand', default=500)
  model.var('K_f', desc='Non-residential capital', default = 1000)
  model.var('K', desc='Capital', default=1500)
  model.var('L', desc='Loans', default = 100)
  model.var('M', desc='Money deposits', default = 100)
  model.var('MO', desc='Mortgages', default = 100)
  model.var('omega', desc='Wage-share', default = 0.5)
  model.var('rl', desc='Interests rates on loans')
  model.var('rmo', desc='Interests rates on mortgages')
  model.var('S_h', desc='Households savings', default = 100)
  model.var('S_f', desc='Firms savings')
  model.var('S_b', desc='Banks savings')
```

```
    model.var('u', desc='Capacity utilization ratio', default=0.7)
    model.var('V_h', desc='Household net financial wealth', default = 600)
    model.var('V_f', desc='Firms net financial wealth', default = 300)
    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', desc='Household disposable income')
    model.var('Z', desc='Autonomous expenditures')

    model.param('alpha', desc='Propensity to consume out of wages', default=0.
    model.param('beta', desc='Expectation adjustment parameter', default=0.2)
    model.param('gamma_F', desc='% of distributed profits', default=0.4) # 0.4
    model.param('gamma_u', desc='Adjustment parameter for the marginal propens
    model.param('gz', desc='Autonomous grouth rate', default=0.05) # 0.02
    model.param('omegapar', desc='Wage-share', default=0.5) # 0.5
    model.param('rm', desc='Interest rates on money deposits', default=0.02) #
    model.param('spread_l', desc='Spread for loans', default=0.01) # 0.01
    model.param('spread_mo', desc='Spread for mortgages', default=0.005) # 0.0
    model.param('un', desc='Normal capacity utilization ratio', default=0.8) #
    model.param('v', desc='Capitl-Output ratio', default=2.5) # 2.5


    # General equations
    model.add('Y = C + I_t') # Eq1
    model.add('I_t = I_f + I_h') # Eq2
    model.add('omega = omegapar') # Eq 3
    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('K = K_HD + K_f') # Eq 8
    model.add('Z = I_h') # Eq 9

    # Household equations
    model.add('YD = W + FD + rm*M(-1) - rmo*MO(-1)') # Eq 10
    model.add('S_h = YD - C') # Eq 11
    #model.add('d(MO) = d(M) - S_h') # Eq 12
    model.add('d(MO) = (1+gz)*I_h(-1)') # Eq 12
    model.add('C = alpha*W') # Eq 13
    #model.add('C = alpha*YD') # Eq 13
    #model.add('C = alpha*W + (1-alpha)*V_h(-1)') # Eq 13
    model.add('V_h = M + K_HD - MO') # Eq 14


    # Firms
    model.add('L = I_f - FU + L(-1)') # Eq 15
    model.add('FT = Y - W') # Eq 16
    model.add('FU = gamma_F*(FT - rl*L(-1))') # Eq 17
    model.add('FD = (1 - gamma_F)*(FT - rl*L(-1))') # 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 co
    model.add('V_f = K_f - L') # Eq 22
    model.add('S_f = FU - I_f') # Eq 23

    # Banks
    model.add('M = (L - L(-1)) + (MO - MO(-1)) + M(-1)') # 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('S_b = rl*L(-1) + rmo*MO(-1) - rm*M(-1)') # Eq 28

    # Residential investment
    model.add('K_HS = K_HD') # Eq 29
    model.add('d(K_HD) = I_h') # Eq 30
    #model.add('I_h = d(MO)') # Eq 31
    model.add('I_h = d(M) - S_h')
    #model.add('I_h = (1+gz)*I_h(-1)') # Eq 31
    model.add('g_Ih = gz') # Eq 32
    return model
```

Solving

```
base = model()
df = SolveSFC(base, time=500)
df.transpose()
```

Evaluating consistenty

```
evaldf = pd.DataFrame({
    'Households' : base.evaluate('M - MO + K_HD - V_h'),
    'Firms' : base.evaluate('K_f - L - V_f'),
    'Banks' : base.evaluate('L + MO - M - V_b'),
    '[Total Financial Wealth - K]' : base.evaluate('V_f + V_h + V_b - K'),
    "Firm's Funds" : base.evaluate('I_f - FU - d(L)'),
    "Housing" : base.evaluate('K_HD - K_HS'),
    "[Saving - Investment]" : base.evaluate('S_f + S_b + S_h - I_h - I_f'),
}, index = ['Sum'])
evaldf
```

Plots

```
ax = df['K_HD'][10:].pct_change().plot(color = "black", title = "Houses grow
ax.set_yticklabels(['{:,.1%}'.format(x) for x in ax.get_yticks()])
plt.xlabel("Period")
sns.despine()
plt.show()
```

```
ax = df[['C', 'I_t']].apply(lambda x: x/df['Y']).plot(kind = 'area', stacked
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()
```

```
ax = df[['W', 'FT']].apply(lambda x: x/df['Y']).plot(kind = 'area', stacked
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()
```

```
f, (ax1, ax2, ax3) = plt.subplots(1, 3)
df[['S_b', 'S_h', 'S_f']].apply(lambda x: x/df['Y']).plot(kind = "area", ax
ax3.axhline(y = 0, ls = "--", color = "black")
ax3.set_yticklabels(['{:,.0%}'.format(x) for x in ax.get_yticks()])
df[['S_b', 'S_h', 'S_f']][1:].apply(lambda x: x/df['K'][1:]).plot(kind = "ar
ax2.axhline(y = 0, ls = "--", color = "black")
ax2.set_yticklabels(['{:,.0%}'.format(x) for x in ax.get_yticks()])
df[['S_b', 'S_h', 'S_f']][1:].plot(kind = "area", ax = ax1, title = "Sector
ax1.axhline(y = 0, ls = "--", color = "black")
sns.despine()
plt.tight_layout()
plt.show()
```

```
ax = df[['V_b', 'V_h', 'V_f']].apply(lambda x: np.abs(x)/df['K']).plot(kind =
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()
```

```
ax = df['Y'][50:].pct_change().plot(color = "black", title = "GDP growth rat
ax.set_yticklabels(['{:,.2%}'.format(x) for x in ax.get_yticks()])
ax.axhline(y = 0, color = "gray", ls="--")
sns.despine()
plt.show()
```

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

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

Analytical solution

```
base = model()
```

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

Model variables

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

Model parameters

```
for i in base.parameters:
  globals()[i] = sp.Symbol(i)
```

Defining equations

General equations

```
Y = _C(t) + _I_t(t)
I = _I_f(t) + _I_h(t)
Yk = _K_f(t-1)/v
u = _Y(t)/_Yk(t)
Z = _I_h(t)
W = omegapar*_Y(t)
K = _K_HD(t) + _K_f(t)
```

Households

```
C = alpha*_YD(t)
YD = _W(t) + _FD(t) + rm*_M(t-1) - _rmo(t)*_MO(t-1)
S_h = _YD(t) - _C(t)
dMO = (_M(t) - _M(t-1)) + _S_h(t)
V_h = _M(t) + _K_HD(t) - _MO(t)
```

Firms

```
I_f = _h(t)*_Y(t)
dK_f = _I_f(t)
L = _I_f(t) - _FU(t) + _L(t-1)
FT = _FU(t) + _FD(t)
FU = gamma_F*(_FT(t) - _rl(t)*_L(t-1))
FD = (1 - gamma_F)*(_FT(t) - _rl(t)*_L(t-1))
h = _h(t-1)*gamma_u*(_u(t)-un) + _h(t-1)
S_f = _FU(t) - _I_f(t)
V_f = _K_f(t) - _L(t)
```

Banks

```
M = (_L(t) - _L(t-1)) + (_MO(t) - _MO(t-1)) + _M(t-1)
rmo = rm + spread_mo
rl = rm + spread_l
V_b = _L(t) + _MO(t) - _M(t)
S_b = _rl(t)*_L(t-1) + _rmo(t)*_MO(t-1) - rm*_M(t-1)
```

Residential Investment

```
K_HS = _K_HD(t)
K_HD = _I_h(t) - _I_h(t-1)
I_h = _MO(t) - _MO(t-1)
g_Ih = gz
```

Rearranging

Level of GDP

```
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(_YD(t), YD)
EqY = EqY.subs(_W(t), W)
EqY = sp.solve(EqY, _Y(t))[0].collect(alpha).collect(omegapar)
solY = EqY
print('Y = ', solY)
print('dY/d alpha = ', EqY.diff(alpha))
print('dY/d omega = ', EqY.diff(omegapar))

print("\nReplacing the initial values.....")
EqY = EqY.subs(_h(t), h)
EqY = EqY.subs(alpha, df.loc[1, 'alpha']).subs(omegapar, df.loc[1, 'omegapar
EqY = EqY.subs(un, df.loc[1, 'un']).subs(gamma_u, df.loc[1, 'gamma_u'])
EqY = EqY.subs(_u(t), df.loc[1, 'u']).subs(_h(t-1), df.loc[1, 'h'])
EqY = EqY.subs(_I_h(t), df.loc[1, 'I_h']).subs(_YD(t), df.loc[1, 'YD'])
EqY = EqY.subs(_M(t-1), df.loc[0, 'M']).subs(_FD(t), df.loc[1, 'FD']).subs(_I
EqY = EqY.subs(rm, df.loc[1, 'rm'])
print('Y0 = ', df.loc[1, 'Y'])
print('Y1 = ', EqY)
print('hat Y - Y1 = ', df.loc[2, "Y"] - EqY)
```

$$Y_{SR} = \frac{-I_h(t)}{\alpha \cdot \omega + h(t) - 1}$$

Rearranging

$$Y_{SR} = \frac{I_h(t)}{1 - \alpha \cdot \omega - h(t)}$$

Capacity output ratio (short-run)

$$\frac{Y}{K(-1)} = \frac{Y}{K(-1)}\frac{Yk}{Yk} = \frac{u}{v}$$
$$\therefore u = \frac{Y}{K}v$$

```
Equ = Y - _Y(t)
Equ = Equ.subs(_C(t), _C(t)*v/_K_f(t-1)).subs(_I_t(t), _I_t(t)*v/_K_f(t-1)).s
Equ = Equ.subs(_C(t), C).subs(_I_t(t), I)
Equ = Equ.subs(_I_f(t), I_f)
Equ = Equ.subs(_W(t), W)
Equ = Equ.expand()
Equ = Equ.subs(_Y(t)/_K(t-1), _u(t)/v)
```

$$i_{h_t} = \frac{I_h}{K_{t-1}}$$

```
i_h = sp.Function('i_h')
Equ = Equ.subs(_I_h(t)/_K(t-1), i_h(t))
Equ = Equ.subs(_Y(t), solY)
Equ = Equ.collect(_u(t))
Equ = Equ.subs(_YD(t), YD)
Equ = Equ.subs(_W(t), W)
Equ = sp.solve(Equ, _u(t))[0].factor().collect(alpha*omegapar).collect(_h(t)
solu = Equ.collect(alpha**2).collect(omegapar).collect(_YD(t)).collect(alpha
Equ = Equ.subs(_Y(t), EqY)
print('u = ', solu)
print('\nDerivatives')
print('du/d alpha = ', Equ.diff(alpha))
print('du/d omega = ', Equ.diff(omegapar))

print("\nReplacing the initial values.....")
Equ = Equ.subs(alpha, df.loc[1,'alpha']).subs(omegapar, df.loc[1,'omegapar']
Equ = Equ.subs(_gk(t-1), df.loc[0,'gk']).subs(_h(t), df.loc[1,'h']).subs(v, (
Equ = Equ.subs(i_h(t), df.loc[1,'I_h']/df.loc[0,'K']).subs(_YD(t), df.loc[1,
Equ = Equ.subs(_I_h(t), df.loc[1,'I_h']).subs(_K_f(t-1), df.loc[0,'K_f'])
Equ = Equ.subs(_M(t-1), df.loc[0, 'M']).subs(_FD(t), df.loc[1, 'FD']).subs(_I
Equ = Equ.subs(rm, df.loc[1, 'rm'])

print('u0 = ', df.loc[1,'u'].round(3))
print('u1 = ', Equ.round(3))
print("hat u - u1 = ", df.loc[1,'u'].round(3) - Equ.round(3))
```

```
print(sp.latex(solu).replace("omegapar", "\\omega"))
```

$$u_{SR} = \frac{v}{\left(\alpha\omega + h(t) - 1\right)\operatorname{K_{f}}\left(t - 1\right)}\left(\alpha^{2}\left(\omega^{2}Y(t) + \omega\left(rmM(t-1) + \operatorname{FD}\left(t\right) - \operatorname{MO}\left(t - 1\right)\operatorname{rmo}\left(t\right)\right)\right) + \alpha\left(\omega\left(\operatorname{I_{h}}\left(t\right) + Y(t)h(t) - Y(t)\right) - rmM(t-1) - \operatorname{FD}\left(t\right) + \operatorname{MO}\left(t - 1\right)\operatorname{rmo}\left(t\right)\right) - \operatorname{I_{h}}\left(t\right)\right)$$

$$u_{SR} = \frac{-v \cdot i_h(t)}{(\alpha \cdot \omega + h(t) - 1)}$$

Rerranging

$$u_{SR} = \frac{v \cdot i_h(t)}{1 - \alpha \cdot \omega - h(t)}$$