calibration

September 30, 2025

1 Calibration procedure (six-steps procedures)

[2]:
```
import pandas as pd
import numpy as np
import agentpy as ap

from model.model import DualEcoModel
from utils.analysis import create_matrices_from_params
from utils.analysis import create_matrices_from_output
```

1.1 initialisation des matrices

[8]:
```
params_1 = {}
model_1 = DualEcoModel(params_1)
model_1.init_params()
create_matrices_from_params(model_1.p)
```

[8]:
```
(         H    F    B    G   CB  sigma
 M      0.0  0.0  0.0  0.0  0.0    0.0
 A      0.0  0.0  0.0  0.0  0.0    0.0
 D      0.0  0.0  0.0  0.0  0.0    0.0
 B      0.0  0.0  0.0  0.0  0.0    0.0
 L      0.0  0.0  0.0  0.0  0.0    0.0
 E      0.0  0.0  0.0  0.0  0.0    0.0
 V     -0.0 -0.0 -0.0 -0.0 -0.0    0.0
 sigma  0.0  0.0  0.0  0.0  0.0    0.0,
           H    F    B    G   CB  sigma
 C       0.0  0.0  0.0  0.0  0.0    0.0
 W       0.0  0.0  0.0  0.0  0.0    0.0
 Z       0.0  0.0  0.0  0.0  0.0    0.0
 T       0.0  0.0  0.0  0.0  0.0    0.0
 iota_A  0.0  0.0  0.0  0.0  0.0    0.0
 iota_B  0.0  0.0  0.0  0.0  0.0    0.0
 iota_L  0.0  0.0  0.0  0.0  0.0    0.0
 iota_D  0.0  0.0  0.0  0.0  0.0    0.0
 Pi_d    0.0  0.0  0.0  0.0  0.0    0.0
 Pi      0.0  0.0  0.0  0.0  0.0    0.0
```

```
DeltaA  0.0  0.0  0.0  0.0  0.0    0.0
DeltaB  0.0  0.0  0.0  0.0  0.0    0.0
DeltaM  0.0  0.0  0.0  0.0  0.0    0.0
DeltaL  0.0  0.0  0.0  0.0  0.0    0.0
DeltaD  0.0  0.0  0.0  0.0  0.0    0.0
DeltaE  0.0  0.0  0.0  0.0  0.0    0.0
L_def   0.0  0.0  0.0  0.0  0.0    0.0
sigma   0.0  0.0  0.0  0.0  0.0    0.0)
```

1.2 calcul des stocks, flux et prix initiaux

[15]:
```
params_2 = {
    'g_ss':0.5,                     # taux de croissance
    'N_E1':5,                       # nombre de entrepreneurs dans le secteur␣
 ↪productif 1
    'N_E2':6,                       # nombre de entrepreneurs dans le secteur␣
 ↪productif 2
    'N_W1':10,                      # nombre de salaries dans le secteur␣
 ↪productif 1
    'N_W2':20,                      # nombre de salaries dans le secteur␣
 ↪productif 2
    'N_WG':40,                      # nombre de salaries dans le secteur public
    'N_U':40,                       # nombre de chomeurs dans le secteur public
    'phi1':1.1,                     # productivite initial du secteur productif␣
 ↪1
    'phi2':1.4,                     # productivite initial du secteur productif␣
 ↪2
    'w1':2.5,                       # salaire initial du secteur productif 1
    'w2':2,                         # salaire initial du secteur productif 2
    'w_G':2,                        # salaire public
    'w_min':2,                      # salaire minimum
    'tau': 0.2,                     # taux d'impots
    'rho': 0.25,                    # politique de dividende
    'm':0.25,                       # taux de marge brute
    'theta_W':0.75,                 # proportion desire de fonds de salaire
    'theta_E':0.75,                 # proportion desire de capitaux bancaire␣
 ↪(rentabilite)
    'theta_M':0.25,                 # proportion desire de liquidite
    'theta_Zbar':0.75,              # proportion reglementaire des allocations␣
 ↪publics
    'r_D': 0.2,                     # taux d'interet sur les depots bancaires
    'r_L': 0.7,                     # taux d'interet sur les credits bancaires
    'r_B': 0.5,                     # taux d'interet sur les bons du tresors
    'r_A': 0.1,                     # taux d'interet sur les avances de la␣
 ↪Banque Centrale
}
```

[16]:
```
model_2 = DualEcoModel(params_2)
model_2.init_params()
model_2.calc_block_1()
model_2.calc_block_2()
model_2.calc_block_3()
model_2.calc_block_4()
create_matrices_from_params(model_2.p, digits=2)
```

0.5681818181818178

[16]:
```
(            H       F      B       G      CB  sigma
 M      240.52  154.50   1.70    0.00 -319.47  77.25
 A        0.00    0.00   0.00    0.00    0.00   0.00
 D      -21.31   28.12  -6.82    0.00    0.00   0.00
 B        0.00    0.00  -2.56 -316.92  319.47   0.00
 L        0.00  -28.12  28.12    0.00    0.00   0.00
 E        0.00  -77.25 -20.45    0.00    0.00 -97.70
 V     -219.21  -77.25  -0.00  316.92   -0.00  20.45
 sigma    0.00    0.00   0.00    0.00    0.00   0.00,
              H       F      B       G      CB  sigma
 C      -111.88  111.88   0.00    0.00    0.00    0.0
 W       169.50  -89.50   0.00  -80.00    0.00    0.0
 Z        60.00    0.00   0.00  -60.00    0.00    0.0
 T       -31.24   -0.00  -2.27   33.51    0.00    0.0
 iota_A    0.00    0.00   0.00    0.00    0.00    0.0
 iota_B    0.00    0.00  -0.85 -105.64  106.49   -0.0
 iota_L    0.00  -13.12  13.12    0.00    0.00    0.0
 iota_D   -2.84    3.75  -0.91    0.00    0.00   -0.0
 Pi_d    -10.48   12.75  -2.27    0.00    0.00    0.0
 Pi        0.00    0.00   0.00  106.49 -106.49    0.0
 DeltaA    0.00    0.00   0.00    0.00    0.00    0.0
 DeltaB    0.00    0.00   0.85  105.64 -106.49    0.0
 DeltaM  -80.17  -25.75  -0.57   -0.00  106.49    0.0
 DeltaL    0.00    9.38  -9.38    0.00    0.00    0.0
 DeltaD    7.10   -9.38   2.27    0.00    0.00   -0.0
 DeltaE    0.00    0.00   0.00    0.00    0.00    0.0
 L_def     0.00    0.00   0.00    0.00    0.00    0.0
 sigma    -0.00    0.00  -0.00    0.00   -0.00   -0.0)
```

1.3 creation des agents et distribution des stocks et flux

[ ]:
```
model = ap.Model()
households = ap.AgentList(model, 10)
households
```

[ ]:
```
owners = households.random(2)
workers = households.random(7)
households.s_U = 1
```

```
households.s_W = 0
households.s_E = 0
owners.s_E = 1
owners.s_U = 0
workers.s_W = 1
workers.s_U = 0
sum(households.s_U), sum(households.s_W), sum(households.s_E)
```

[ ]:
```
households.x = 0
subworkers = workers.to_list().random(5)
subworkers.x = 5
sum(households.x)
```

[ ]:
```
sum_params(model2.p, 'N_E')
```

[ ]:
```
def create_households(m):
    p = m.p
    bank_owners = ap.AgentList(m, 1)
    bank_owners.s_EB = 1
    bank_owners.s_E = 1
    m.households = bank_owners

    firm_owners1 = ap.AgentList(m, p['N_E1'])
    firm_owners1.s_EB = 0
    firm_owners1.s_E = 1
    m.households += firm_owners1

    firm_owners2 = ap.AgentList(m, p['N_E2'])
    firm_owners2.s_EB = 0
    firm_owners2.s_E = 1
    m.households += firm_owners2

    firm_owners3 = ap.AgentList(m, p['N_E3'])
    firm_owners3.s_EB = 0
    firm_owners3.s_E = 1
    m.households += firm_owners3

    # firm_owners = firm_owners1 + firm_owners2 + firm_owners3
    # owners = bank_owners + firm_owners
    # print(len(firm_owners), len(owners))
    # print(len(owners.select(owners.s_EB==0).select(owners.s_E==1)))
    # print(list(owners.select(owners.s_EB==0).select(owners.s_E==1)))

    private_workers1 = ap.AgentList(m, p['N_W1'])
    private_workers1.s_EB = 0
    private_workers1.s_E = 0
    m.households += private_workers1
```

```
    private_workers2 = ap.AgentList(m, p['N_W2'])
    private_workers2.s_EB = 0
    private_workers2.s_E = 0
    m.households += private_workers2

    private_workers3 = ap.AgentList(m, p['N_W3'])
    private_workers3.s_EB = 0
    private_workers3.s_E = 0
    m.households += private_workers3

    # private_workers = private_workers1 + private_workers2 + private_workers3

    public_workers = ap.AgentList(m, p['N_WG'])
    public_workers.s_EB = 0
    public_workers.s_E = 0
    m.households += public_workers
    # workers = private_workers + public_workers
    # print(list(private_workers1))

    unemployed = ap.AgentList(m, p['N_U'])
    unemployed.s_EB = 0
    unemployed.s_E = 0
    m.households += unemployed

    # m.households = owners + workers + unemployed
    # print(len(households.select(households.s_EB==0).select(households.
 ↪s_E==1)))
    # print(list(households.select(households.s_EB==0).select(households.
 ↪s_E==1)))
    return m
```

[ ]:
```
model2 = create_households(model2)
households = model2.households
p2 = model2.p
print('Num households', sum_params(p2, 'N') + 1, len(households))
print('Num bank owners', 1, len(households.select(households.s_EB == 1)))
print('Num firm owners', sum_params(p2, 'N_E'), len(households.
 ↪select(households.s_EB==0).select(households.s_E==1)))
print('Num firm owners', sum_params(p2, 'N_E'), len(households.
 ↪select(households.s_EB==0).select(households.s_E==1)))
```

[ ]:
```
a1 = ap.AgentList(model, 2)
a2 = ap.AgentList(model, 5)
a1 + a2
```

[ ]:
```

```

[ ]: