SAM

June 24, 2024

[1]:
```
import sys
sys.path.append(r'C:\Users\USER\Documents\GEModelTools-master')
sys.path.append(r'C:
↪\Users\USER\Documents\GitHub\GEModelToolsNotebooks\HANK-SAM')
```

[2]:
```
%load_ext autoreload
%autoreload 2

import pickle
import numpy as np
import scipy.optimize as optimize

import matplotlib.pyplot as plt
colors = plt.rcParams['axes.prop_cycle'].by_key()['color']
plt.rcParams.update({"axes.grid" : True, "grid.color": "black", "grid.alpha":"0.
↪25", "grid.linestyle": "--"})
plt.rcParams.update({'font.size': 14})

from HANKSAMModel import HANKSAMModelClass
```

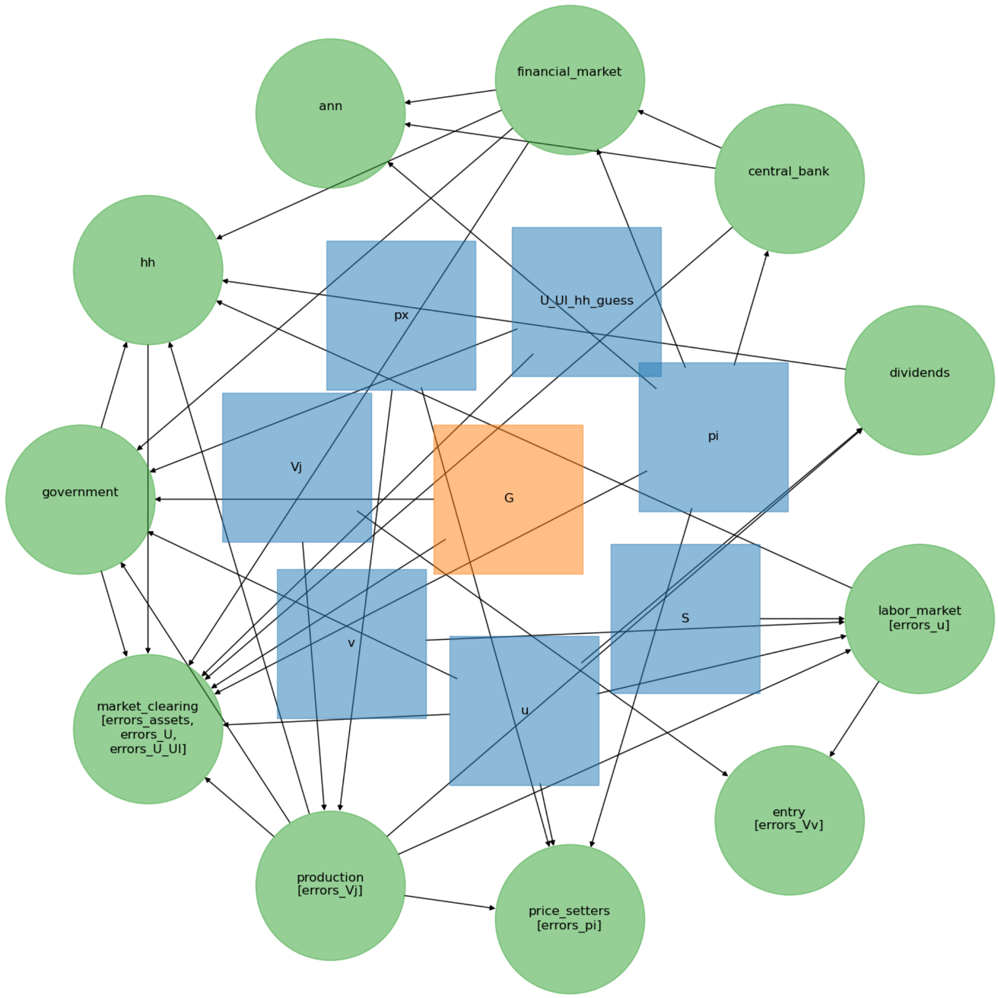
[3]:
```
def create_fig(figsize=(6,6/1.5)):

    fig = plt.figure(figsize=figsize,dpi=100)
    ax = fig.add_subplot(1,1,1)

    return fig,ax

def format_fig(fig,ax,ylabel='',T_max=48,legend=True):

    if legend: ax.legend(frameon=True)
    ax.set_xlabel('months')
    ax.set_ylabel(ylabel)
    ax.set_xticks(np.arange(T_max+1)[::12])
    ax.set_xlim([0,T_max]);

    fig.tight_layout()
```

[4]:
```
model = HANKSAMModelClass(name='baseline')
```

[5]:
```
model.info()
```

```
settings:
 par.py_hh = False
 par.py_blocks = False
 par.full_z_trans = True
 par.warnings = True
 par.T = 480

households:
 grids_hh: [a]
 pols_hh: [a]
 inputs_hh: [w,r,tau,div,transfer]
 inputs_hh_z: [delta,lambda_u_s]
 outputs_hh: [a,c,u_ALL,u_UI]
 intertemps_hh: [vbeg_a]

aggregate:
 shocks: [G]
 unknowns: [px,Vj,v,u,S,pi,U_UI_hh_guess]
 targets:
[errors_Vj,errors_Vv,errors_u,errors_pi,errors_assets,errors_U,errors_U_UI]

blocks (inputs -> outputs):
 production: [px,Vj] -> [w,TFP,delta,errors_Vj]
 labor_market: [v,S,delta,u] -> [theta,lambda_v,lambda_u_s,errors_u]
 entry: [Vj,lambda_v] -> [errors_Vv]
 price_setters: [px,pi,TFP,u] -> [errors_pi]
 central_bank: [pi] -> [i]
 dividends: [TFP,u,w] -> [div]
 financial_market: [pi,i] -> [q,r]
 government: [G,U_UI_hh_guess,w,u,q] -> [Phi,transfer,X,taut,tau,taxes,B]
 hh: [delta,div,lambda_u_s,r,tau,transfer,w] -> [A_hh,C_hh,U_ALL_hh,U_UI_hh]
 market_clearing: [G,TFP,pi,i,C_hh,u,q,B,U_ALL_hh,U_UI_hh_guess,U_UI_hh,A_hh] ->
[Y,clearing_Y,qB,errors_assets,errors_U,errors_U_UI]
 ann: [i,r,pi] -> [i_ann,r_ann,pi_ann]
```

[36]:
```
model.draw_DAG(figsize=(15,15),order=['shocks','unknowns','blocks'])
# Save the plot to the specified path with 600 DPI
output_path = r"C:\Users\USER\Documents\GitHub\SAM Korea\DAG.png"
plt.savefig(output_path, dpi=600)

plt.show()
```



[7]:
```python
import pandas as pd

# Using raw string literals
df = pd.read_csv(r'C:\Users\USER\Documents\GitHub\SAM Korea\sam.csv',␣
 ↪index_col=0)
print(df)
```

```
                  w         r        tau       chi  transfer       div
t
2000Q1  1865.000000  7.447143  38.077683  1.480000      13.6  8.503000
2000Q2  1923.750000  6.607268  37.859068  1.437500      13.7  8.042500
2000Q3  1982.500000  5.767394  37.640453  1.395000      13.9  7.582000
2000Q4  2041.250000  4.927520  37.421838  1.352500      13.7  7.121500
```

```
2001Q1  2100.000000  4.087645  37.203224  1.310000      13.5  6.661000
…                …         …          …         …         …
2022Q4  9505.000000  1.219082  53.717909  0.950000      14.3  3.729500
2023Q1  9620.000000  0.632449  53.947629  0.950000      14.8  3.816000
2023Q2  9022.367345  1.406032  55.672217  1.053420      13.1  1.354002
2023Q3  9121.229685  1.430053  55.881505  1.051857      13.3  1.344988
2023Q4  9220.629617  1.440693  56.085259  1.050412      13.0  1.330373

[96 rows x 6 columns]
```

[8]:
```
par = model.par
ss = model.ss
path = model.path
```

[9]:
```
import pandas as pd
df = pd.read_csv(r'C:\Users\USER\Documents\GitHub\SAM Korea\sam.csv',␣
 ↪index_col=0)
#w,r,tau,div,transfer
model.ss.w = df['w']
model.ss.r = df['r']
model.ss.tau = df['tau']
model.ss.div = df['div']
model.transfer = df['transfer']
```

[10]:
```
print(model.ss.w)
print(model.ss.r)
print(model.ss.div)
print(model.ss.tau)
print(model.ss.transfer)
```

```
t
2000Q1    1865.000000
2000Q2    1923.750000
2000Q3    1982.500000
2000Q4    2041.250000
2001Q1    2100.000000
               …
2022Q4    9505.000000
2023Q1    9620.000000
2023Q2    9022.367345
2023Q3    9121.229685
2023Q4    9220.629617
Name: w, Length: 96, dtype: float64
t
2000Q1    7.447143
2000Q2    6.607268
2000Q3    5.767394
2000Q4    4.927520
```

```
2001Q1    4.087645
             …
2022Q4    1.219082
2023Q1    0.632449
2023Q2    1.406032
2023Q3    1.430053
2023Q4    1.440693
Name: r, Length: 96, dtype: float64
t
2000Q1    8.503000
2000Q2    8.042500
2000Q3    7.582000
2000Q4    7.121500
2001Q1    6.661000
             …
2022Q4    3.729500
2023Q1    3.816000
2023Q2    1.354002
2023Q3    1.344988
2023Q4    1.330373
Name: div, Length: 96, dtype: float64
t
2000Q1    38.077683
2000Q2    37.859068
2000Q3    37.640453
2000Q4    37.421838
2001Q1    37.203224
             …
2022Q4    53.717909
2023Q1    53.947629
2023Q2    55.672217
2023Q3    55.881505
2023Q4    56.085259
Name: tau, Length: 96, dtype: float64
nan
```

[12]:
```
model.find_ss(do_print=True)
```

```
par.A = 0.3680
par.kappa = 1.8883
ss.w = 0.7500
ss.delta = 0.0200
ss.lambda_u_s = 0.3000
ss.lambda_v = 0.5000
ss.theta = 0.6000
ss.u = 0.0625
ss.S = 0.0625
household problem in ss solved in 5.0 secs [4082 iterations]
```

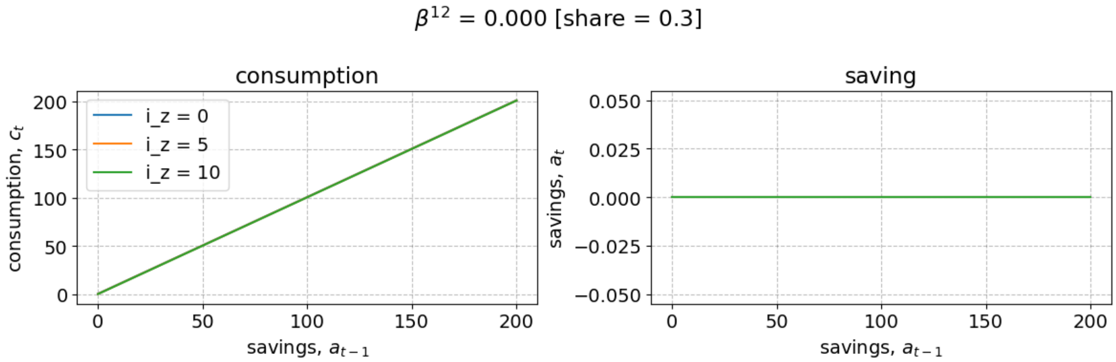
```
household problem in ss simulated in 0.3 secs [1959 iterations]
ss.G = 0.4226
ss.clearing_Y = 0.0000
par.jump_G = 0.0042
steady state found in 5.3 secs
```

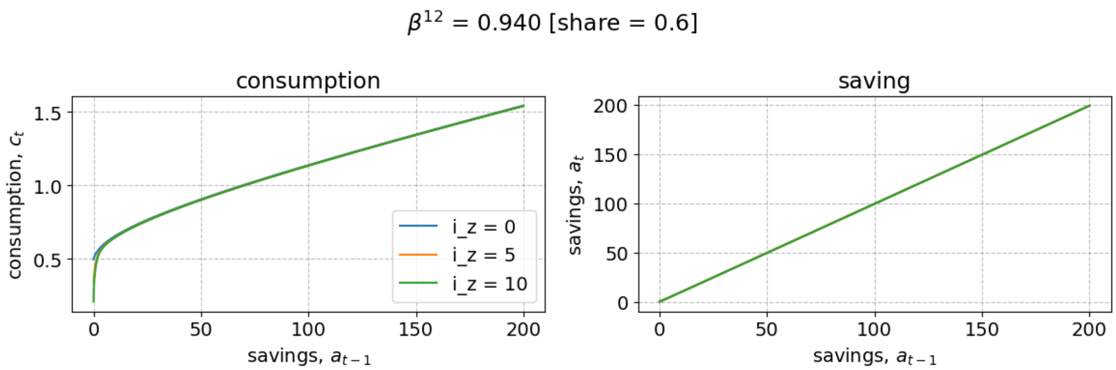
[13]:
```
model.test_ss()
```

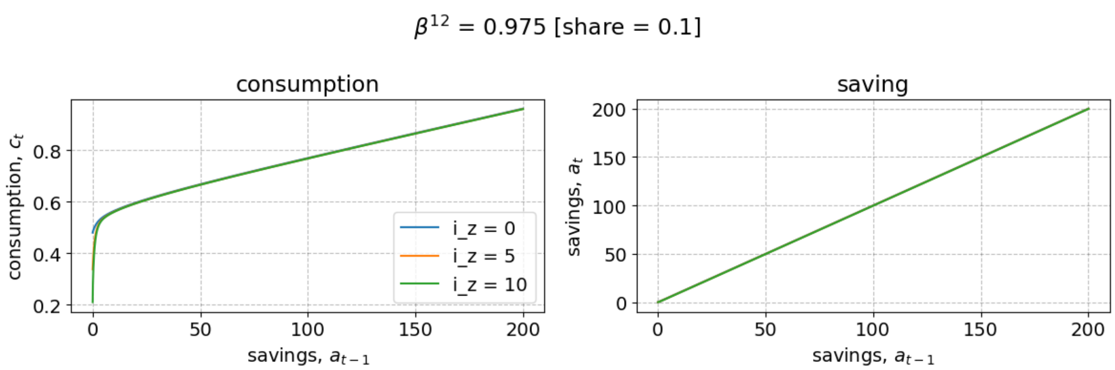
```
w              :       0.7500
TFP            :       1.0000
px             :       0.8333
delta          :       0.0200
Vj             :       3.7766
errors_Vj      :       0.0000
v              :       0.0375
S              :       0.0625
u              :       0.0625
theta          :       0.6000
lambda_v       :       0.5000
lambda_u_s     :       0.3000
errors_u       :       0.0000
errors_Vv      :       0.0000
pi             :       0.0000
errors_pi      :       0.0000
i              :       0.0017
div            :       0.2344
q              :      33.9797
r              :       0.0017
G              :       0.4226
U_UI_hh_guess  :       0.0551
Phi            :       0.0312
transfer       :      -0.2344
X              :       0.2194
taut           :       0.3000
tau            :       0.3000
taxes          :       0.2203
B              :       0.0161
A_hh           :       0.5482
C_hh           :       0.5149
U_ALL_hh       :       0.0625
U_UI_hh        :       0.0551
Y              :       0.9375
clearing_Y     :       0.0000
qB             :       0.5482
errors_assets  :       0.0000
errors_U       :       0.0000
errors_U_UI    :       0.0000
i_ann          :       0.0200
```

```
r_ann          :      0.0200
pi_ann         :      0.0000
```

[14]:
```
for i_fix in range(par.Nfix):

    fig = plt.figure(figsize=(12,4),dpi=100)
    a_max = 500

    # a. consumption
    I = par.a_grid < a_max

    ax = fig.add_subplot(1,2,1)
    ax.set_title(f'consumption')

    for i_z in [0,par.Nz//2,par.Nz-1]:
        ax.plot(par.a_grid[I],ss.c[i_fix,i_z,I],label=f'i_z = {i_z}')

    ax.legend(frameon=True)
    ax.set_xlabel('savings, $a_{t-1}$')
    ax.set_ylabel('consumption, $c_t$')

    # b. saving
    I = par.a_grid < a_max

    ax = fig.add_subplot(1,2,2)
    ax.set_title(f'saving')

    for i_z in [0,par.Nz//2,par.Nz-1]:
        ax.plot(par.a_grid[I],ss.a[i_fix,i_z,I],label=f'i_z = {i_z}')

    ax.set_xlabel('savings, $a_{t-1}$')
    ax.set_ylabel('savings, $a_{t}$')

    fig.suptitle(fr'$\beta^{{12}}$ = {par.beta_grid[i_fix]**12:.3f} [share =␣
↪{par.beta_shares[i_fix]}]')
    fig.tight_layout()

# Save the plot to the specified path with 600 DPI
output_path = r"C:\Users\USER\Documents\GitHub\SAM Korea\plot.png"
plt.savefig(output_path, dpi=600)

plt.show()
```

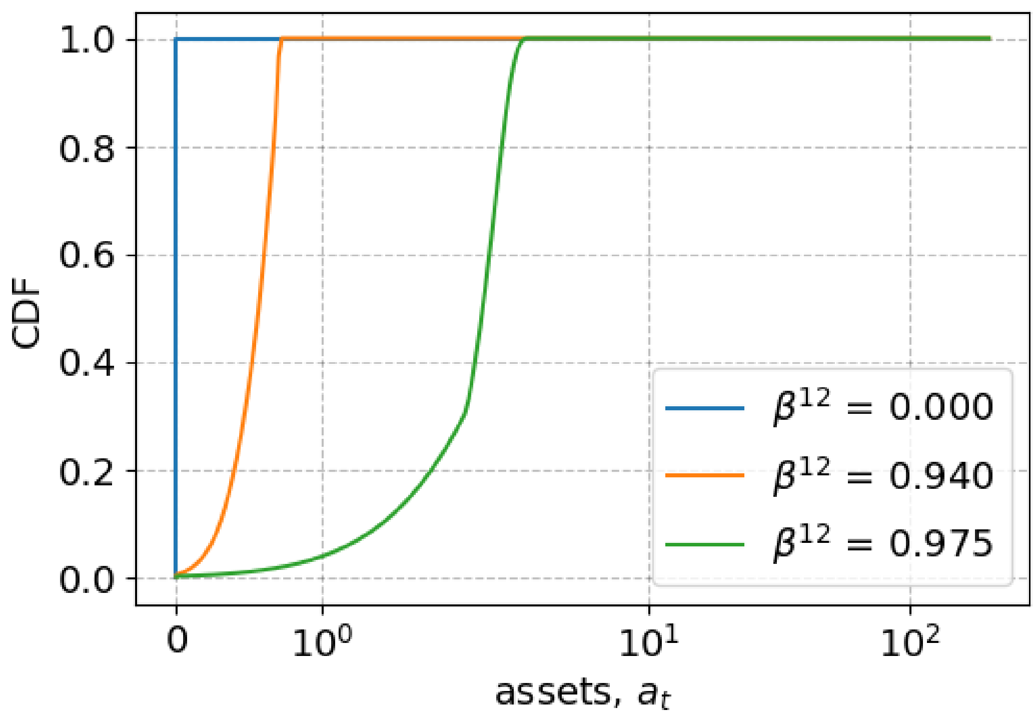






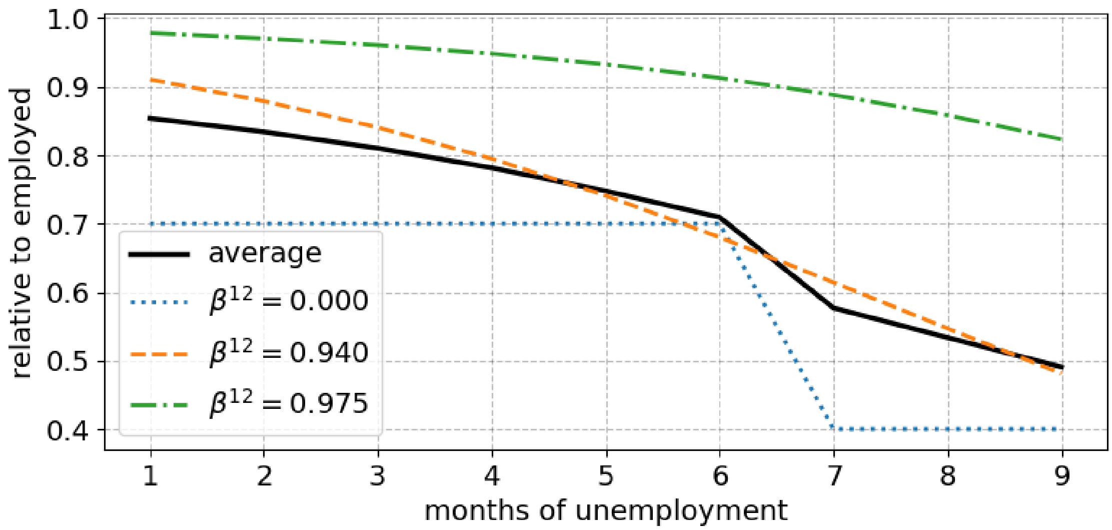
[15]:
```
fig = plt.figure(figsize=(6,4),dpi=100)
ax = fig.add_subplot(1,1,1)

for i_fix in range(par.Nfix):
    y = np.insert(np.cumsum(np.sum(ss.D[i_fix],axis=0)),0,0.0)
    ax.plot(np.insert(par.a_grid,0,par.a_grid[0]),y/y[-1],
            label=f'$\\beta^{{12}}$ = {par.beta_grid[i_fix]**12:.3f}')
```

```
ax.legend(frameon=True)
ax.set_xlabel('assets, $a_{t}$')
ax.set_ylabel('CDF')
ax.set_xscale('symlog')
# Save the plot to the specified path with 600 DPI
output_path = r"C:\Users\USER\Documents\GitHub\SAM Korea\plot.png"
plt.savefig(output_path, dpi=600)

plt.show()
```



[16]:
```
fig,ax = create_fig(figsize=(8,6/1.5))

# baseline
C_e,C_u,C_u_dur = model.calc_Cs()
ax.plot(np.arange(1,par.Nu),C_u_dur[:-1]/C_e,ls='-',lw=2.
↪5,color='black',label='average')

for i_fix,ls in zip(range(par.Nfix),[':','--','-.']):
    C_e,C_u,C_u_dur = model.calc_Cs(i_fix=i_fix)
    ax.plot(np.arange(1,par.Nu),C_u_dur[:-1]/
↪C_e,lw=2,ls=ls,label=f'$\\beta^{{12}} = {par.beta_grid[i_fix]**12:.3f}$')
```

```
# details
ax.set_xticks(np.arange(1,par.Nu));
ax.set_xlabel('months of unemployment')
ax.set_ylabel('relative to employed')
ax.legend(frameon=True,ncol=1)

fig.tight_layout()
# Save the plot to the specified path with 600 DPI
output_path = r"C:\Users\USER\Documents\GitHub\SAM Korea\duration.png"
plt.savefig(output_path, dpi=600)

plt.show()
```



[17]:
```
C_e,C_u,C_u_dur = model.calc_Cs()
C_drop_ss = (C_u/C_e-1)*100
C_drop_ex = (C_u_dur[6]-C_u_dur[5])/((1-ss.tau)*(par.phi_obar-par.phi_ubar)*ss.
  ↪w)*100
print(f'{C_drop_ss = :.2f}')
print(f'{C_drop_ex = :.2f}')
```

```
C_drop_ss = -22.01
C_drop_ex = -43.90
```

[18]:
```
model.test_path()
```

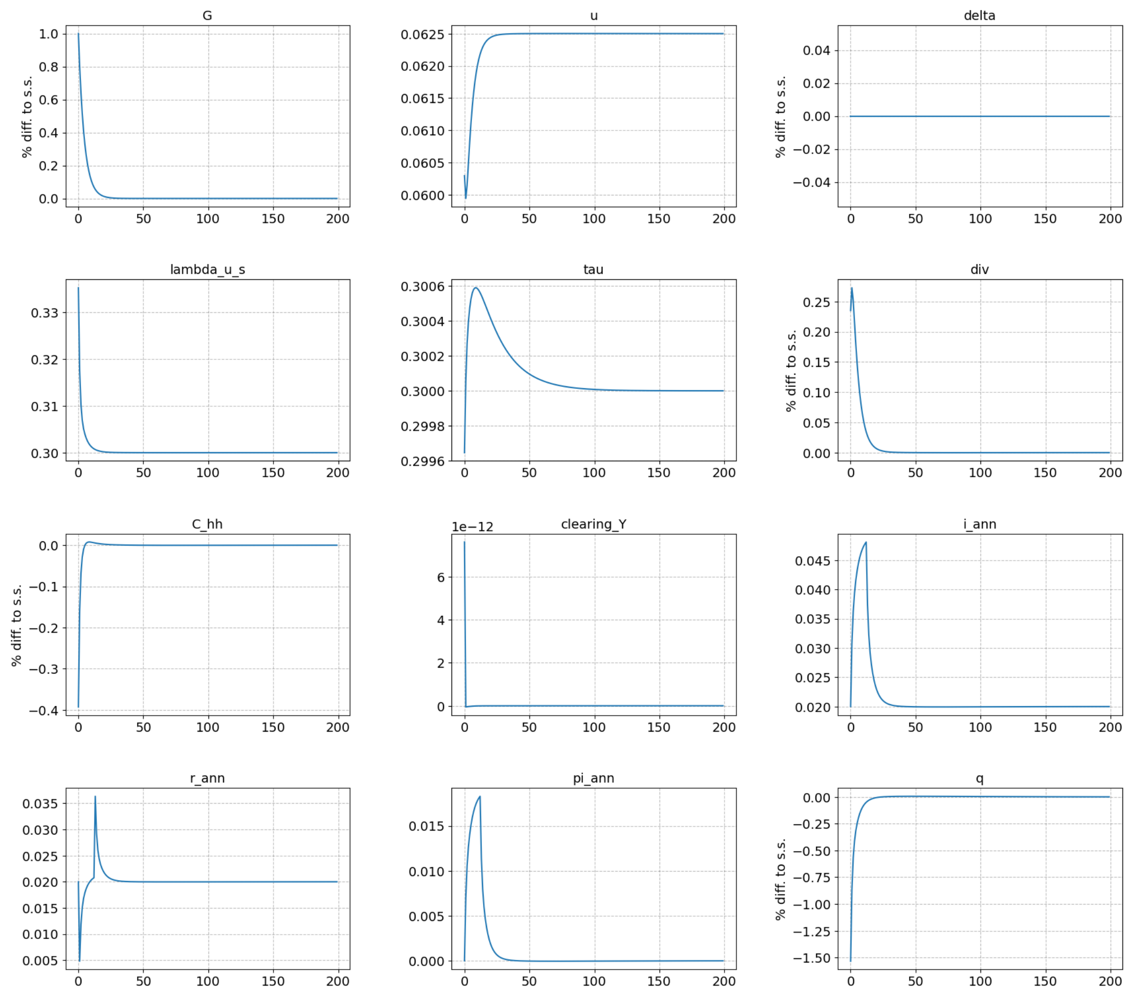
```
shocks: G
unknowns: px Vj v u S pi U_UI_hh_guess

look at max(abs(path.VARNAME[:]-ss.VARNAME)):
```

```
blocks.production
 w                 0.0e+00
 TFP               0.0e+00
 delta             0.0e+00
 errors_Vj         0.0e+00 [target]
blocks.labor_market
 theta             0.0e+00
 lambda_v          0.0e+00
 lambda_u_s        0.0e+00
 errors_u          0.0e+00 [target]
blocks.entry
 errors_Vv         0.0e+00 [target]
blocks.price_setters
 errors_pi         0.0e+00 [target]
blocks.central_bank
 i                 0.0e+00
blocks.dividends
 div               0.0e+00
blocks.financial_market
 q                 0.0e+00
 r                 2.2e-16
blocks.government
 Phi               0.0e+00
 transfer          0.0e+00
 X                 0.0e+00
 taut              0.0e+00
 tau               0.0e+00
 taxes             0.0e+00
 B                 0.0e+00
hh
 A_hh              7.9e-10
 C_hh              9.0e-12
 U_ALL_hh          1.3e-16
 U_UI_hh           1.2e-16
blocks.market_clearing
 Y                 0.0e+00
 clearing_Y        9.0e-12
 qB                0.0e+00
 errors_assets     7.9e-10 [target]
 errors_U          1.1e-15 [target]
 errors_U_UI       1.2e-16 [target]
blocks.ann
 i_ann             4.4e-16
 r_ann             6.7e-16
 pi_ann            0.0e+00
```

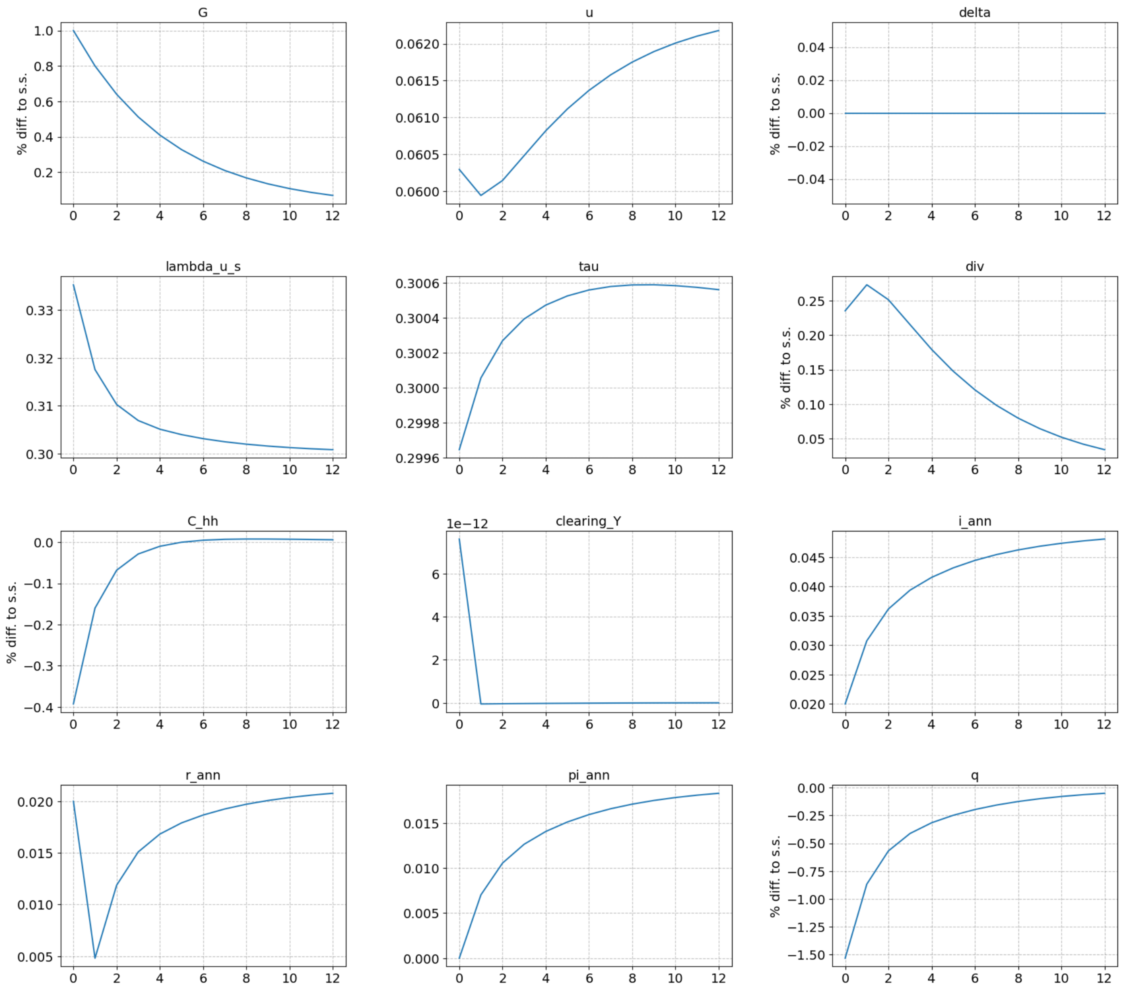
[23]:
```
model.compute_jacs(do_print=False,skip_shocks=True)
```

[24]:
```
model.find_transition_path(shocks=['G'],do_print=False,do_end_check=False)
```

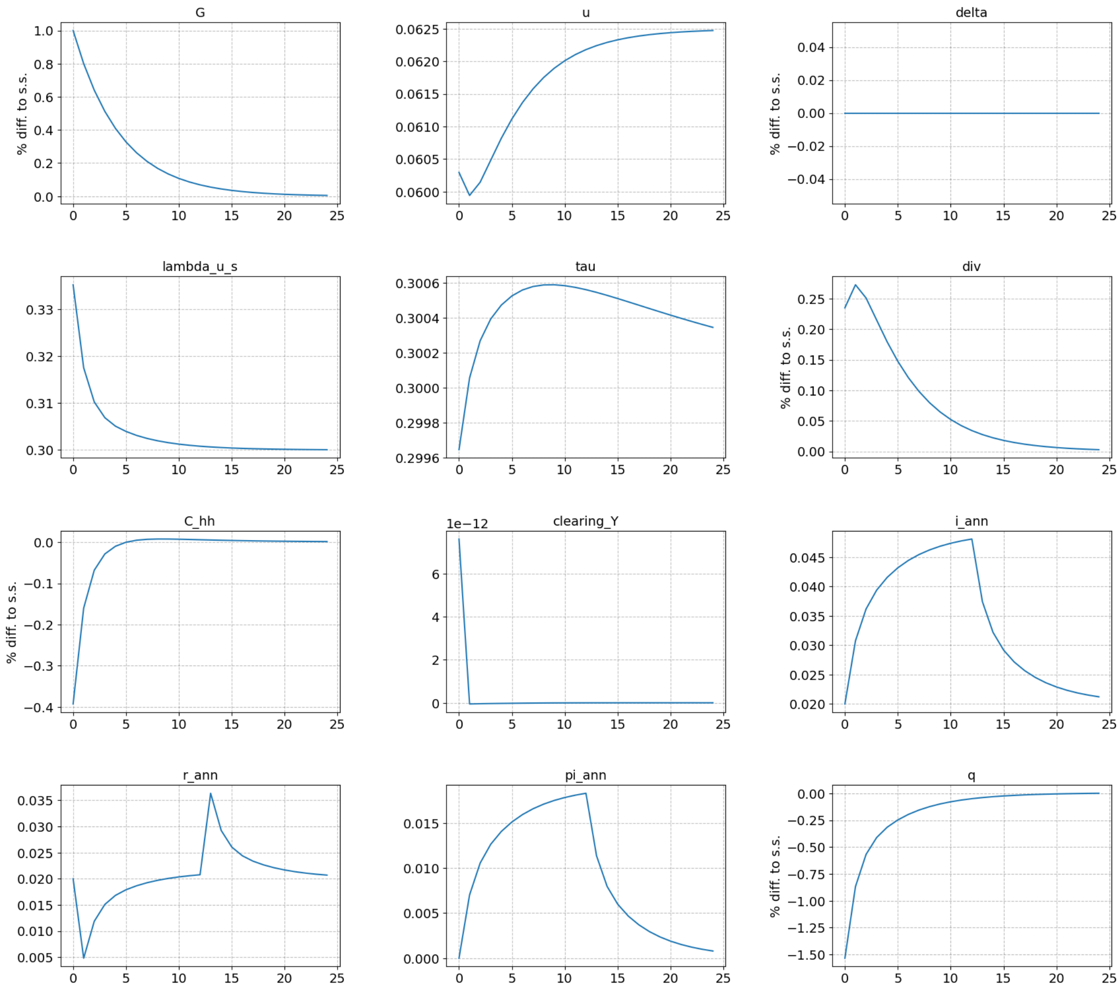
[25]:
```
paths =␣
↪['G','u','delta','lambda_u_s','tau','div','C_hh','clearing_Y','i_ann','r_ann','pi_ann','q']
lvl_value = ['u','lambda_u_s','tau','i_ann','r_ann','pi_ann','clearing_Y']
model.
↪show_IRFs(paths,lvl_value=lvl_value,T_max=200,ncols=3,do_shocks=False,do_targets=False)
```



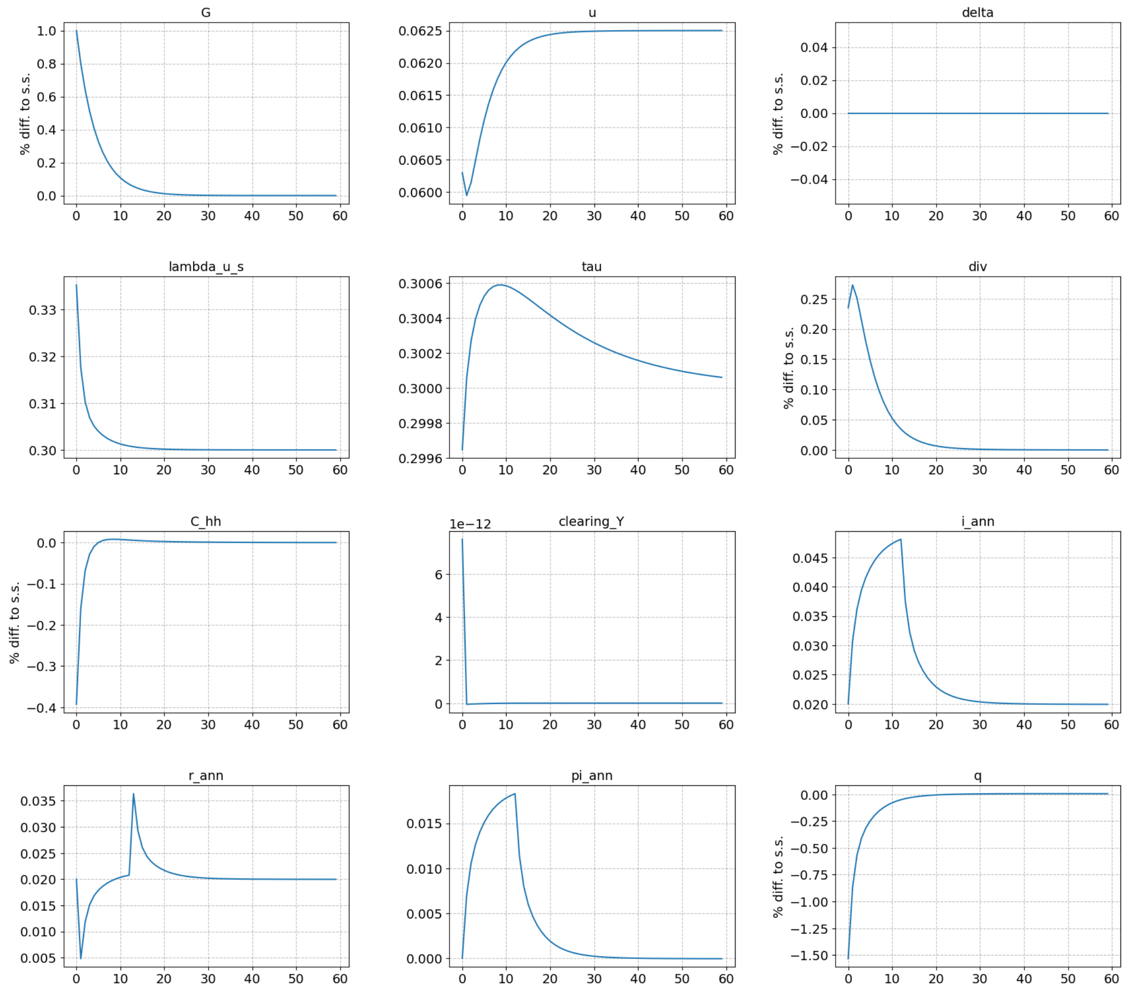
[26]:
```
paths =␣
↪['G','u','delta','lambda_u_s','tau','div','C_hh','clearing_Y','i_ann','r_ann','pi_ann','q']
lvl_value = ['u','lambda_u_s','tau','i_ann','r_ann','pi_ann','clearing_Y']
model.
↪show_IRFs(paths,lvl_value=lvl_value,T_max=13,ncols=3,do_shocks=False,do_targets=False)
```



[33]:
```
paths =␣
↪['G','u','delta','lambda_u_s','tau','div','C_hh','clearing_Y','i_ann','r_ann','pi_ann','q']
lvl_value = ['u','lambda_u_s','tau','i_ann','r_ann','pi_ann','clearing_Y']
model.
↪show_IRFs(paths,lvl_value=lvl_value,T_max=25,ncols=3,do_shocks=False,do_targets=False)
```



[34]:
```
paths =
 ↪['G','u','delta','lambda_u_s','tau','div','C_hh','clearing_Y','i_ann','r_ann','pi_ann','q']
lvl_value = ['u','lambda_u_s','tau','i_ann','r_ann','pi_ann','clearing_Y']
model.
 ↪show_IRFs(paths,lvl_value=lvl_value,T_max=60,ncols=3,do_shocks=False,do_targets=False)
```

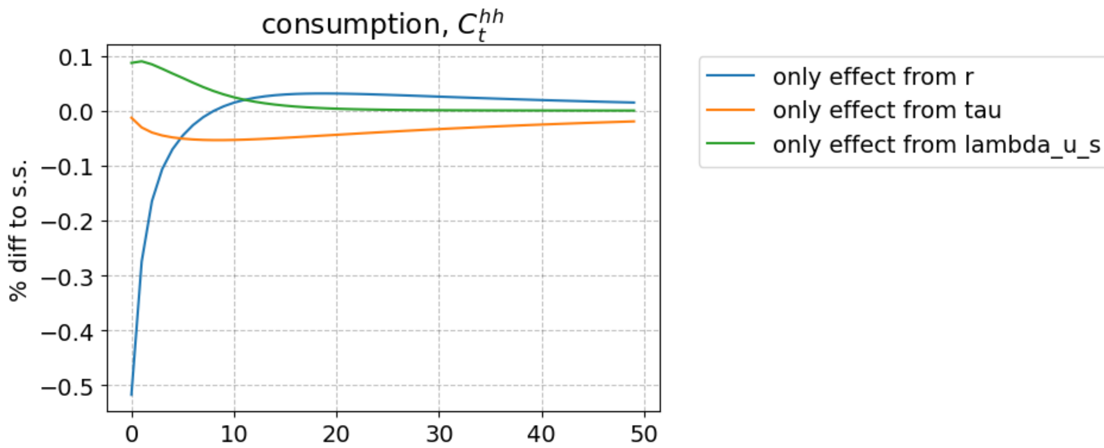


[37]:

```python
T_max = 50

fig_C = plt.figure(figsize=(6,4),dpi=100)
ax_C = fig_C.add_subplot(1,1,1)
ax_C.set_title('consumption, $C_t^{hh}$')

i_color = 0
for use_inputs in [[x] for x in ['r','tau','lambda_u_s']]:

    # a. compute
    print(use_inputs)
    path_alt = model.decompose_hh_path(do_print=True,use_inputs=use_inputs)
    print('')

    # b. plot
```

```
    if use_inputs is None:
        label = 'no inputs'
        ls = '--'
        color = 'black'
    elif use_inputs == 'all':
        label = 'all inputs'
        ls = '-'
        color = 'black'
    else:
        label = f'only effect from {use_inputs[0]}'
        ls = '-'
        color = colors[i_color]
        i_color += 1

    ax_C.plot((path_alt.C_hh[:T_max]/ss.
 ↪C_hh-1)*100,ls=ls,color=color,label=label);

for ax in [ax_C]:
    ax.set_ylabel('% diff to s.s.')
    lgd = ax.legend(frameon=True,ncol=1,bbox_to_anchor=(1.05,1), loc='upper␣
 ↪left',)

# Save the plot to the specified path with 600 DPI
output_path = r"C:\Users\USER\Documents\GitHub\SAM Korea\decom.png"
plt.savefig(output_path, dpi=600)

plt.show()
```

```
['r']
household problem solved along transition path in 0.7 secs
household problem simulated along transition in 0.1 secs

['tau']
household problem solved along transition path in 0.7 secs
household problem simulated along transition in 0.1 secs

['lambda_u_s']
household problem solved along transition path in 0.7 secs
household problem simulated along transition in 0.1 secs
```


consumption, C_t^{hh}
% diff to s.s.
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
0
10
20
30
40
50
only effect from r
only effect from tau
only effect from lambda_u_s

```
[ ]:
```