HANK with Housing Block

- By William Du (JHU) and Jamie Lenney (Bank of England)

We will be running code in live time so we need to import some libraries

We will be running code in live time so we need to import some libraries

In [1]:

```
import numpy as np
from numba import njit
import matplotlib.pyplot as plt
from copy import deepcopy
import importlib
import pickle

from sequence_jacobian import grids, interpolate,simple, create_model, solved
from sequence_jacobian.blocks.stage_block import StageBlock
import warnings
warnings.filterwarnings('ignore')
```

Households

- Each period, households choose to be either owners or renters
- Owners can borrow.
- One asset consumption saving problem
- Idiosyncratic income shocks

The Household problem

Stage 1: Enter period t and draw idiosyncratic labor productivity shock

Labor productivity follows a discretized AR(1) process in logs

$$\log z_t = \rho_z \log z_{t-1} + \epsilon_t$$
$$\epsilon_t \sim \mathcal{N}(0, \sigma_z^2)$$

Stage 1: Enter period t and draw idiosyncratic labor productivity shock

Labor productivity follows a discretized AR(1) process in logs

$$\log z_t = \rho_z \log z_{t-1} + \epsilon_t$$
$$\epsilon_t \sim \mathcal{N}(0, \sigma_z^2)$$

$$V_t^{(1)}(h_{t-1}, z_{t-1}, a_{t-1}) = \Pi_z V_t^{(2)}(h_{t-1}, z_t, a_{t-1})$$

Stage 3: Aggregate shocks are realized and households make discrete choice

- If owner $h = 1$, otherwise renter $h = 0$

Stage 3: Aggregate shocks are realized and households make discrete choice

- If owner $h = 1$, otherwise renter $h = 0$
- Owners choose to either sell house $h = 0$ or continue owning $h = 1$

Stage 3: Aggregate shocks are realized and households make discrete choice

- If owner $h = 1$, otherwise renter $h = 0$
- Owners choose to either sell house $h = 0$ or continue owning $h = 1$
- If renters, choose to either buy house $h = 1$ or keep renting $h = 0$

Stage 3: Aggregate shocks are realized and households make discrete choice

- If owner $h = 1$, otherwise renter $h = 0$
- Owners choose to either sell house $h = 0$ or continue owning $h = 1$
- If renters, choose to either buy house $h = 1$ or keep renting $h = 0$

$$V_t^{(2)}(h_{t-1}, z_t, a_{t-1}) = \max_{h_t \in \{0,1\}} V_t^{(3)}(h_t, z_t, a_{t-1}) + \underbrace{\varepsilon(h_t)}_{\text{taste shock}}$$

Stage 4: Solve consumption saving problem

Stage 4: Solve consumption saving problem

$$V_t^{(3)}(h_t, z_t, a_{t-1}) = \max_{c_t, a_t} u(c_t, h_t) + \beta \mathbb{E}_t V_{t+1}^{(1)}(h_t, z_t, a_t)$$

Owners Budget Constraints

If owner choosing to continue owning,

$$
\begin{aligned}
c_t + a_t &= (1 + r_t) a_{t-1} + y_t z_t \\
a_t &\geq \min\{-\kappa_h P_t, a_{t-1}\}
\end{aligned}
$$

Owners Budget Constraints

If owner choosing to continue owning,

$$
\begin{aligned}
c_t + a_t &= (1 + r_t) a_{t-1} + y_t z_t \\
a_t &\geq \min\{-\kappa_h P_t, a_{t-1}\}
\end{aligned}
$$

If owner choosing to sell, and become a renter

$$
\begin{aligned}
c_t + a_t &= (1 + r_t) a_{t-1} + y_t z_t + P_t - \xi - p_{r,t} \\
a_t &\geq 0
\end{aligned}
$$

Renters Budget Constraints

If renter looking to continue renting

$$
\begin{aligned}
c_t + a_t &= (1 + r_t) a_{t-1} + y_t z_t - p_{r,t} \\
a_t &\geq 0
\end{aligned}
$$

Renters Budget Constraints

If renter looking to continue renting

$$
\begin{aligned}
c_t + a_t &= (1 + r_t)a_{t-1} + y_t z_t - p_{r,t} \\
a_t &\geq 0
\end{aligned}
$$

If renter looking to buy a house

$$
\begin{aligned}
c_t + a_t &= (1 + r_t)a_{t-1} + y_t z_t - P_t - \xi \\
a_t &\geq -\kappa_h P_t
\end{aligned}
$$

General Equilibrium

General Equilibrium

- Standard New Keynesian setup
    - Sticky prices
    - Sticky wages
- Government taxes labor income to pay off interest on government debt.
    - Tax schedule is determined by a standard fiscal rule.
- Inertial Taylor rule

Housing and rental markets

There is a fixed supply of houses $\bar{H}$.

Houses are either owned by homeowners or by the rental sector.

$$\bar{H} = H_o + H_r$$

Rental Sector

- There is a rental sector that borrows from a bank to buy houses H_r and converts them into rental units at rate $\psi > 1$.
    - Rental units are rented out to households at rent price p_r.

$$H_{seg} = \psi H_r$$

where H_{seg} is the supply of rental properties.

The rental sector makes zero profits in expectation.

$$p_{r,t} = \frac{1}{\psi}\mathbb{E}_t\left[r_{t+1}P_t + P_t - P_{t+1}\right]$$

And consumes all unexpected profit

$$C_{rental} = p_{r,t}H_{seg,t} + \frac{1}{\psi}(P_t - P_{t-1})H_{seg,t-1} - \frac{1}{\psi}r_t P_{t-1}H_{seg,t-1}$$

Market Clearing

Asset market

- In aggregate, households deposit A_t at the bank.
- The bank uses these deposits to loan to the rental sector and buy government bonds B_t

$$A_t = P_t H_r + B_t$$

Asset market

- In aggregate, households deposit A_t at the bank.
- The bank uses these deposits to loan to the rental sector and buy government bonds B_t

$$A_t = P_t H_r + B_t$$

Goods market

$$Y_t = C_t + C_{rental,t}$$

Bringing the model to code

In [2]:

```
# Import the households
from python.hhproblem_WD import hh, cali
import python.hhproblem_WD as hhprob_WD_st
```

Household calibration

- Key calibration targets:
    - Median house price to median earnings: 32 times quarterly income
    - Share of households that are homeowners : 64%
    - Median rental price as a proportion of household income: 26%
    - Transactional costs of buying a house: 1% of the house price
    - Loan to house price: 75%

Specifying General Equilibrium Blocks into SSJ toolkit

In [3]:

```
# Some general equilibrium equations

@simple
def rentprice(rb,ph,relHR):
    pr = ((rb(+1)*ph+ph-ph(+1))*relHR  ) # arbitrage equation for the rental company, asset pricing equation
    return pr

@simple
def mkt_clearing(AD, NE, C, L, Y, B, mu,hours,Div,ph,HR,Eph,beta,pr,rb,HOO,HOR,HRO,transac,relHR,hbar):

    asset_mkt = AD - B - ph*HR*relHR # bank issues deposits to rental companies who buy rental housing off them

    housing_mkt=hbar-relHR*HR-HOO

    crent=pr*HR+(ph-ph(-1))*HR(-1)*relHR-rb*ph(-1)*HR(-1)*relHR # rental companies consumption

    labor_mkt = hours*NE - L

    goods_mkt = Y - C -crent -transac*(HOR+HRO)

    return asset_mkt, goods_mkt,labor_mkt,crent,housing_mkt
```

In [4]:

```
# Remaining Equations

@solved(unknowns={'i':0.005},targets=['i_res'],solver='brentq')
def Taylor(i,pi, rstar, phi,rhor,epsr):
    i_res=(1-rhor)*(rstar+phi*pi)+rhor*(i(-1))+epsr-i # Taylor Rule
    return i_res


@simple
def Fisher(i,pi):
    rb=(1+i(-1))/(1+pi)-1
    r=rb
    return rb,r

@simple
def wageinflation(w,pi):
    piw=w-w(-1)+pi
    return piw

@simple
def firm(Y, Z, pi, mu, kappa,w):
    L = Y/Z
    Div = Y - w * L #- mu/(mu-1)/(2*kappa) * (1+pi).apply(np.log)**2 * Y
    return Div,L

@solved(unknowns={'B': 5.6,'Tax': 0.02 }, targets=['B_res','Tax_res'], solver="brentq")
def fiscal(Tax,B,rb,gammatax,rhotax,Taxss,Bss,w,L,Y,Yss,gammay):
    B_res=B(-1)*(1+rb)-Tax*w*L-B

    Tax_res = Taxss*((Tax(-1)/Taxss)**rhotax)*((B/Bss)**(gammatax*(1-rhotax)))*((Y/Yss)**(gammay*(1-rhotax)))-Tax
    return B_res,Tax_res

@simple
def Agglabsupply(w,wss,frisch,beta,phil,piw,hourss,kappaw):
    hours=(frisch*((w/wss).apply(np.log)+ 1/kappaw*(piw-beta*piw(+1)))+hourss.apply(np.log)).apply(np.exp)
    return hours

@simple
def expectations(ph):
    Eph=ph(+1)
    return Eph

@simple # dynamic philips curve
def nkpc(pi, w, Z, Y, r, mu, kappa):
    nkpc_res = kappa * (w / Z - 1 / mu) + Y(+1) / Y * (1 + pi(+1)).apply(np.log) / (1 + r(+1))\
               - (1 + pi).apply(np.log)
    return nkpc_res

@simple
def TaylorSS(pi, rstar, phi):
    i=rstar+phi*pi # Taylor Rule
    return i

@simple
```

```
def fiscalSS(rb, B,w,L):
    Tax = (rb * B(-1))/(w*L)
    return Tax

@simple
def AgglabsupplySS(w,frisch,L,Ladj):
    hours=L/Ladj
    phil=w/hours**(1/frisch)
    return phil,hours

@solved(unknowns={'q': 0.2/0.005}, targets=['eq_arb'], solver="brentq")
def eqarb(q,rb,Div):
    eq_arb=1+rb-(Div(+1)+q(+1))/q
    return eq_arb

@simple
def fpricesSS(Div,rb,relHR,phss):
    q=Div/rb
    ph=phss
    pr=rb*ph*relHR
    return q, pr,ph

@simple
def nkpc_ss(Z, mu):
    w = Z / mu
    return w
```

Solve for Steady State

Solve for Steady State

In [5]:

```
%%time
blocks_ss = [hh, firm, TaylorSS,Fisher, fiscalSS,AgglabsupplySS,fpricesSS, mkt_clearing,nkpc_ss,expectations]

hank_ss = create_model(blocks_ss, name="One-Asset HANK SS")

unknowns_ss = {'beta': [0.975,0.985],'vh2':[0,0.4]}

targets_ss = {'asset_mkt': 0,'housing_mkt': 0}

ss0 = hank_ss.solve_steady_state(cali, unknowns_ss, targets_ss, solver="broyden_custom")
```

```
CPU times: user 13.9 s, sys: 85.3 ms, total: 14 s
Wall time: 14 s
```

Household transitions

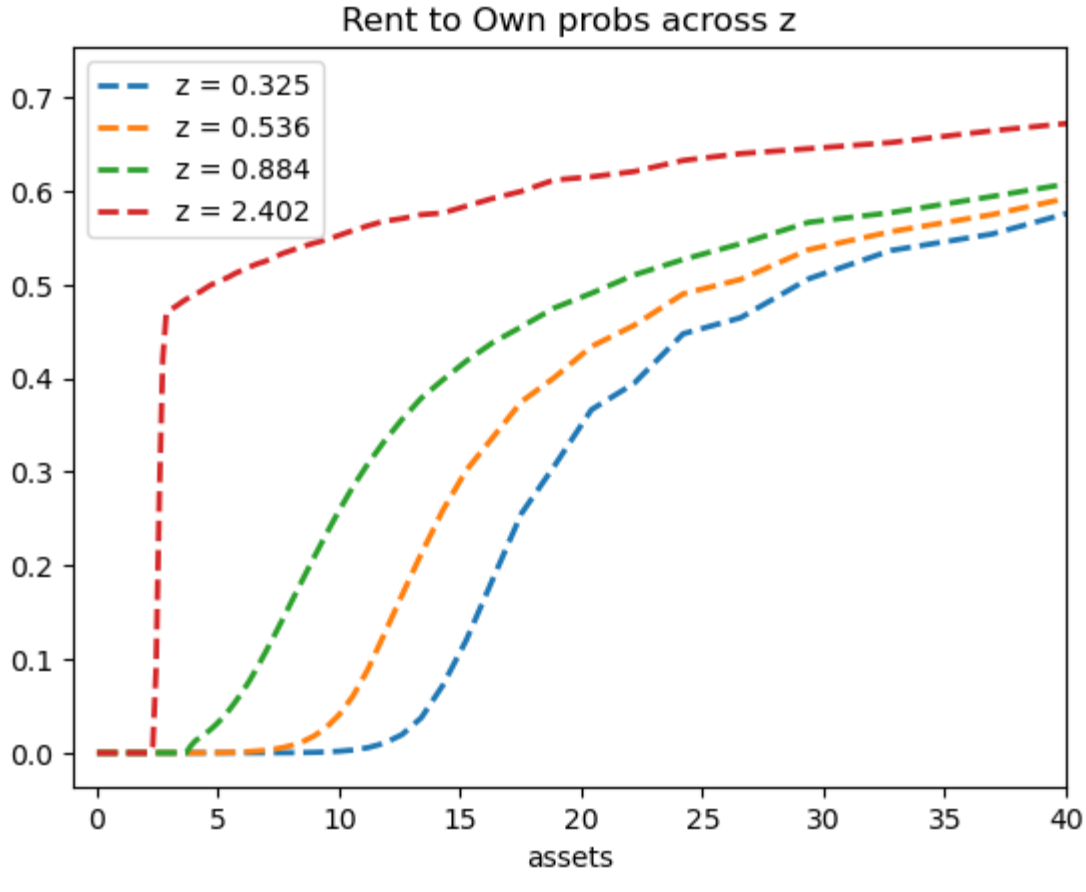
Household transitions

In [6]:

```
ss0['hourss']=ss0['hours']
ss0['HRSS']=ss0['HR']

P_transition = ss0.internals['hh']['house_choice']['law_of_motion'].P # transition matrix between states across assets and
a_grid_r = ss0.internals['hh']['a_grid_r'] # renters asset grid
a_grid_o = ss0.internals['hh']['a_grid_o'] # owners asset grid
z_grid = ss0.internals['hh']['z_grid'] # idiosyncratic prod. grid
```

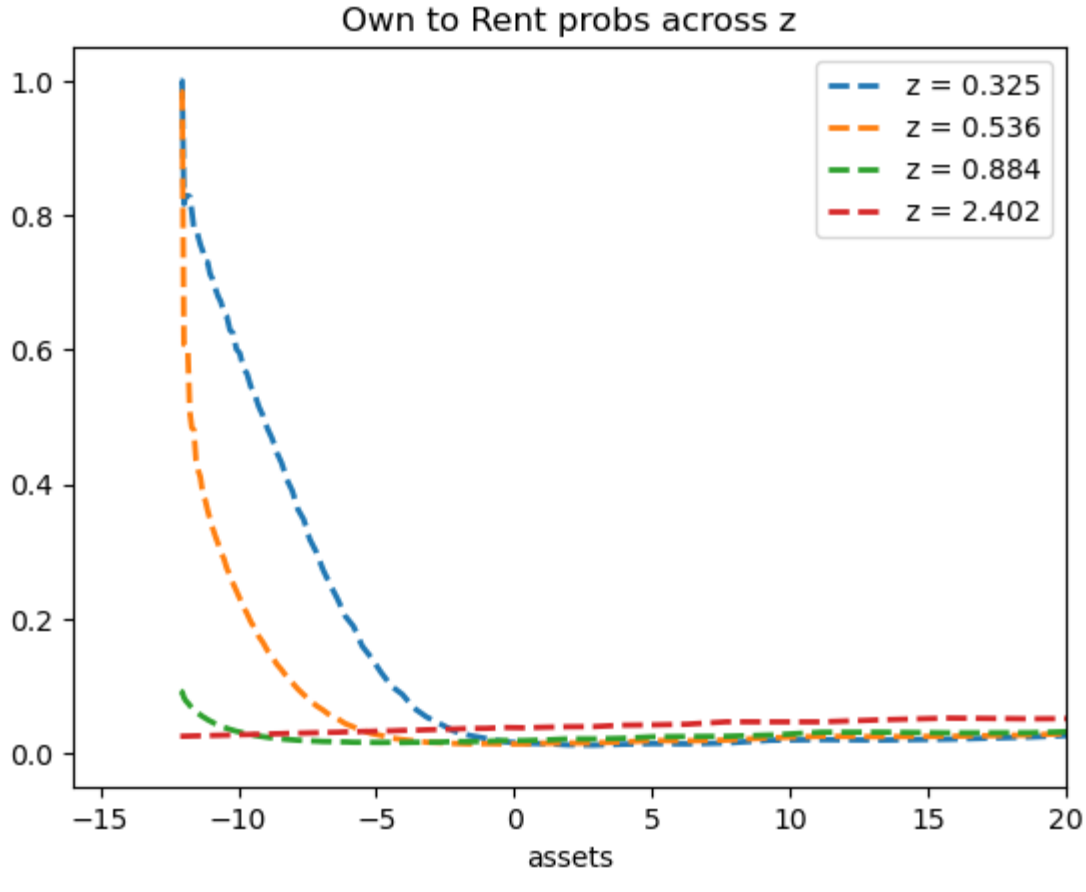
In [7]:

```
plt.plot(a_grid_r, P_transition[2, 3, 0], linewidth=2, linestyle = '--' , label = 'z = ' + str(round(z_grid[0],3) )) #
plt.plot(a_grid_r, P_transition[2, 3, 1], linewidth=2, linestyle = '--' , label = 'z = ' +str(round(z_grid[1],3))) #
plt.plot(a_grid_r, P_transition[2, 3, 2], linewidth=2, linestyle = '--' , label ='z = ' +str(round(z_grid[2],3))) #
plt.plot(a_grid_r, P_transition[2, 3, 4], linewidth=2, linestyle = '--' , label ='z = ' +str(round(z_grid[4],3))) #
plt.title('Rent to Own probs across z')
plt.legend()
plt.xlim(- 1,40)
plt.xlabel('assets')
plt.show()
```



In [8]:

```
plt.plot(a_grid_o, P_transition[1, 0, 0], linewidth=2, linestyle = '--' , label = 'z = ' + str(round(z_grid[0],3) )) #
plt.plot(a_grid_o, P_transition[1, 0, 1], linewidth=2, linestyle = '--' , label = 'z = ' +str(round(z_grid[1],3))) #
plt.plot(a_grid_o, P_transition[1, 0, 2], linewidth=2, linestyle = '--' , label ='z = ' +str(round(z_grid[2],3))) #
plt.plot(a_grid_o, P_transition[1, 0, 4], linewidth=2, linestyle = '--' , label ='z = ' +str(round(z_grid[4],3))) #
plt.title('Own to Rent probs across z')
plt.legend()
plt.xlim(-ss0['ph']*ss0['kappah'] - 4,20)
plt.xlabel('assets')
plt.show()
```



Computing the het agent Jacobians in the code

Computing the het agent Jacobians in the code

In [9]:

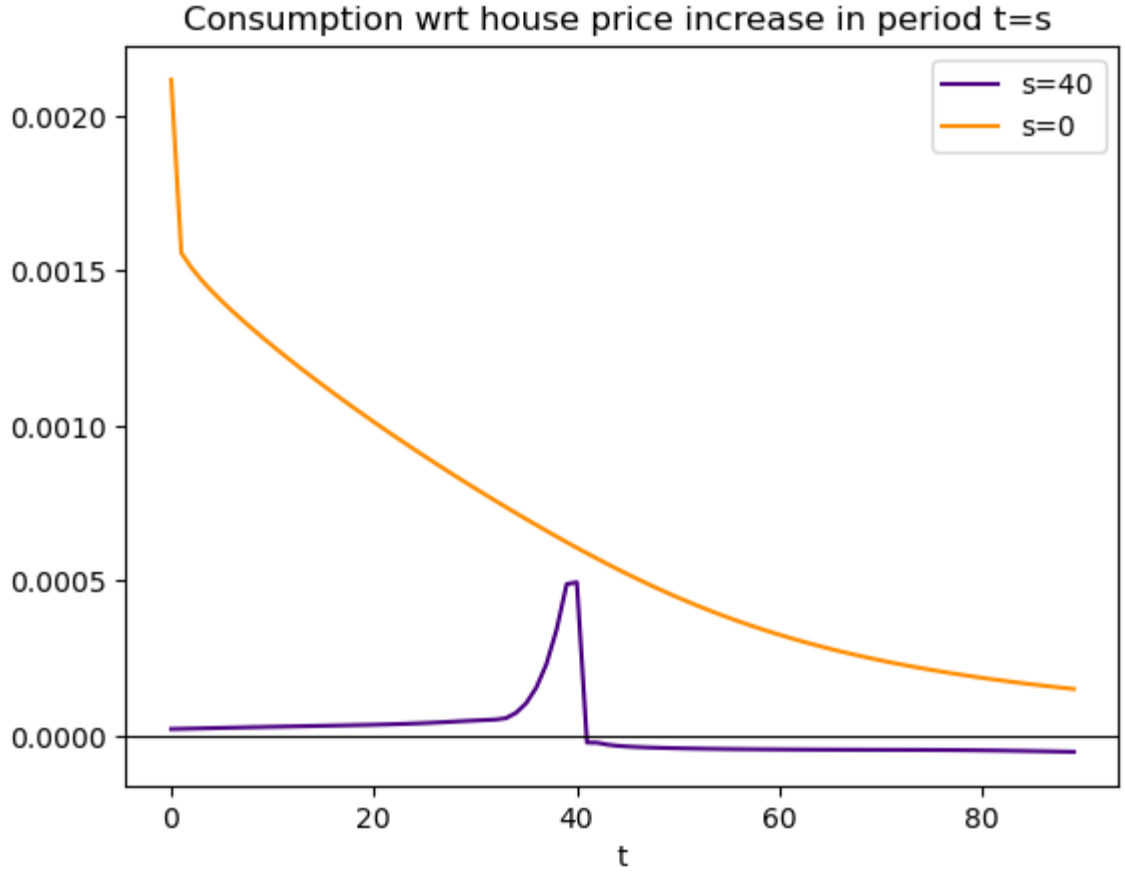
```
%%time
T = 300
J_hh = hh.jacobian(ss0, inputs=['Tax','Div', 'r','w','hours','pr','ph', 'MPC'], T=T)
J_hh_base = deepcopy(J_hh)
```

```
CPU times: user 5.8 s, sys: 3.33 s, total: 9.14 s
Wall time: 3.45 s
```

Jacobians with respect to increase in house price

In [10]:

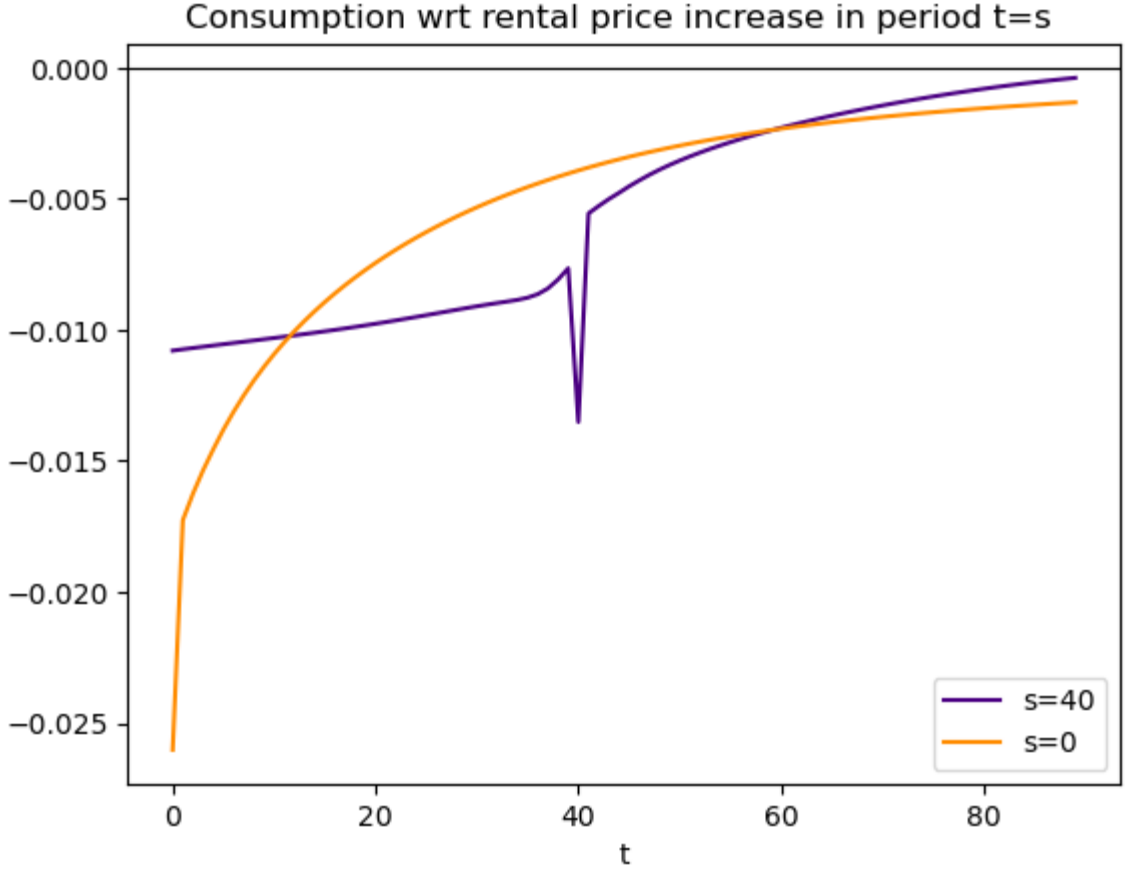
```
plt.plot( J_hh['C']['ph'][:90, 40], color = 'indigo', label = 's=40')
plt.plot( J_hh['C']['ph'][:90, 0], color = 'darkorange', label = 's=0')
plt.xlabel('t')
plt.axhline(0, color='k', linestyle='-', linewidth = .8)
plt.title('Consumption wrt house price increase in period t=s ')
plt.legend()
plt.show()
```



Jacobians with respect to increase in rental price

In [11]:

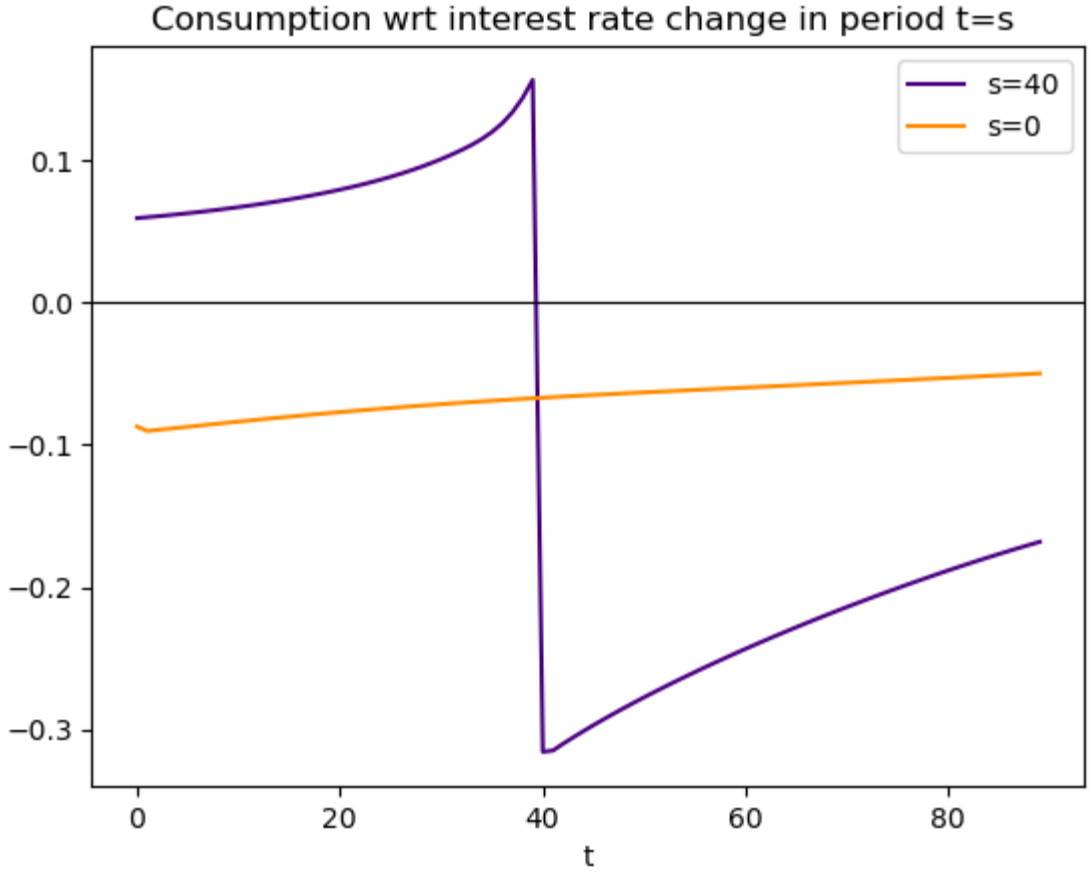
```
plt.plot( J_hh['C']['pr'][:90, 40], color = 'indigo', label = 's=40')
plt.plot( J_hh['C']['pr'][:90, 0], color = 'darkorange', label = 's=0')
plt.xlabel('t')
plt.axhline(0, color='k', linestyle='-', linewidth = .8)
plt.title('Consumption wrt rental price increase in period t=s ')
plt.legend()
plt.show()
```



Jacobians with respect to fall in real interest rate

In [12]:

```
plt.plot( -J_hh['C']['r'][:90, 40], color = 'indigo', label = 's=40')
plt.plot( -J_hh['C']['r'][:90, 0], color = 'darkorange', label = 's=0')
plt.xlabel('t')
plt.axhline(0, color='k', linestyle='-', linewidth = .8)
plt.title('Consumption wrt interest rate change in period t=s ')
plt.legend()
plt.show()
```

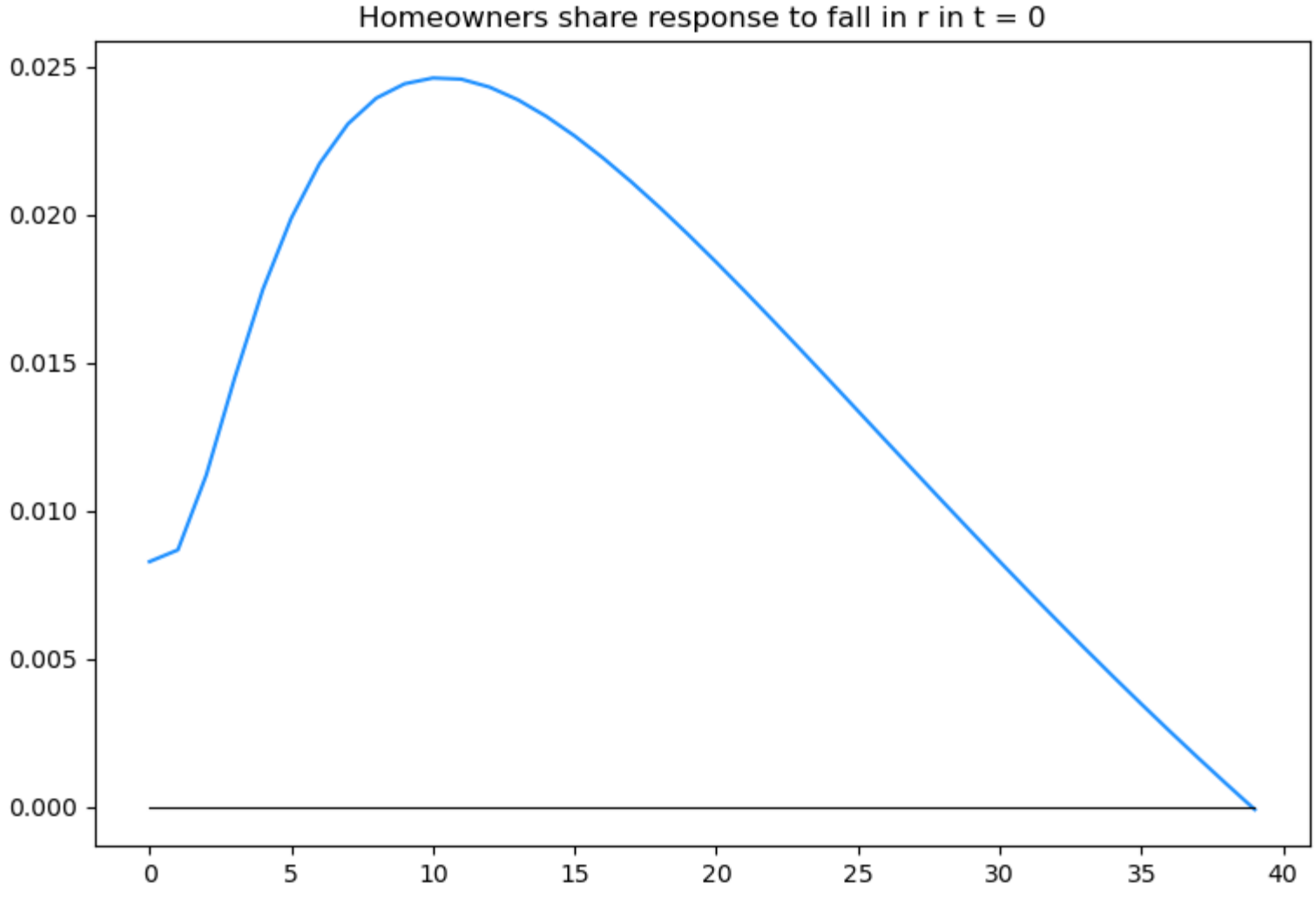


In [13]:

```
fig, axs = plt.subplots(1, figsize=(9, 6))
axs.plot(-J_hh['HOO']['r'][:40, 0] , color = 'dodgerblue')
axs.plot(np.zeros(40), color = 'k', linewidth = .8)
axs.set_title('Homeowners share response to fall in r in t = 0')
```

Out[13]:

```
Text(0.5, 1.0, 'Homeowners share response to fall in r
in t = 0')
```


Homeowners share response to fall in r in t = 0
0.025
0.020
0.015
0.010
0.005
0.000
0
5
10
15
20
25
30
35
40

General equilibrium impulse responses

General equilibrium impulse responses

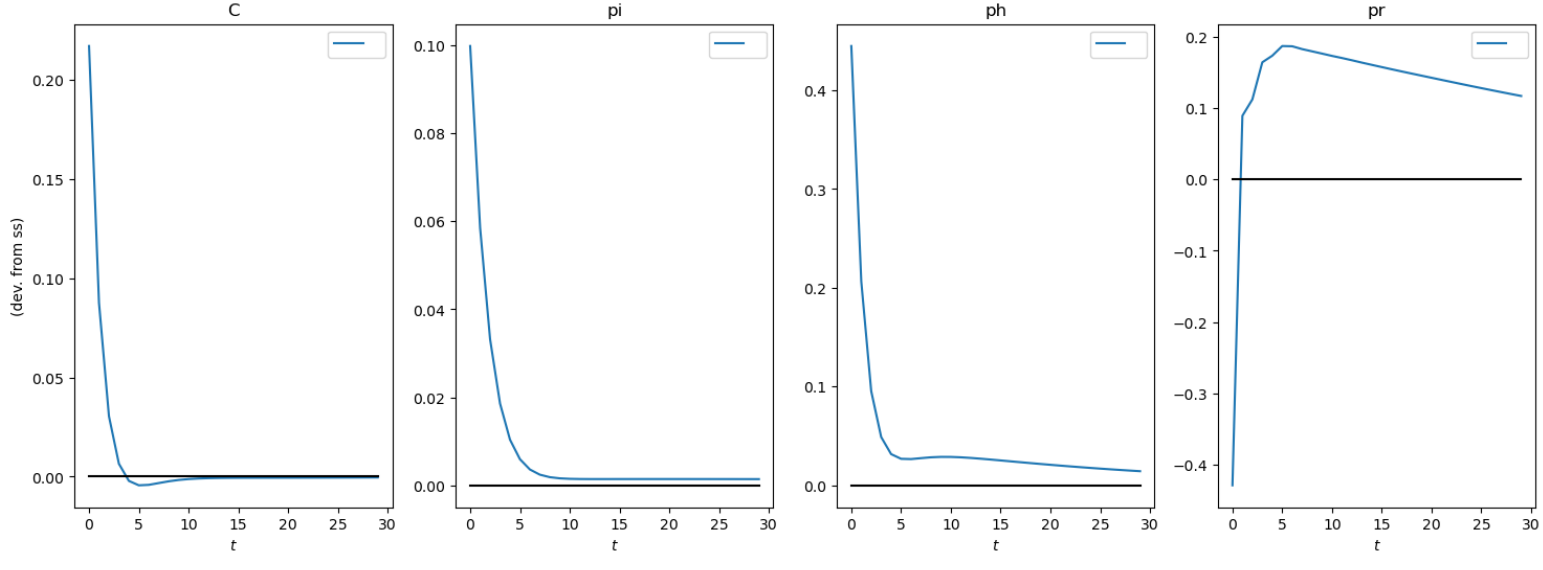
In [14]:

```
%%time
show_irfs = hhprob_WD_st.show_irfs # function to plot Irfs

exogenous = ['epsr', 'Z']
unknowns = [ 'pi','Y','w','ph']
targets = ['nkpc_res','labor_mkt', 'asset_mkt','housing_mkt']

drstar=np.zeros(T)
drstar[0]= -0.0025

rstar_shock_path = {"epsr": drstar}
blocks = [hh, firm, Taylor,Fisher, fiscal,Agglabsupply, mkt_clearing,rentprice,eqarb, nkpc,wageinflation,expectations]
hank = create_model(blocks, name="One-Asset HANK")

IPRs = hank.solve_impulse_linear(ss0, unknowns, targets, rstar_shock_path)
```

```
CPU times: user 17.5 s, sys: 8.97 s, total: 26.5 s
Wall time: 8.55 s
```

In [15]:

```
show_irfs([IPRs],ss0,[ 'C' , 'pi' , 'ph', 'pr'  ])
```

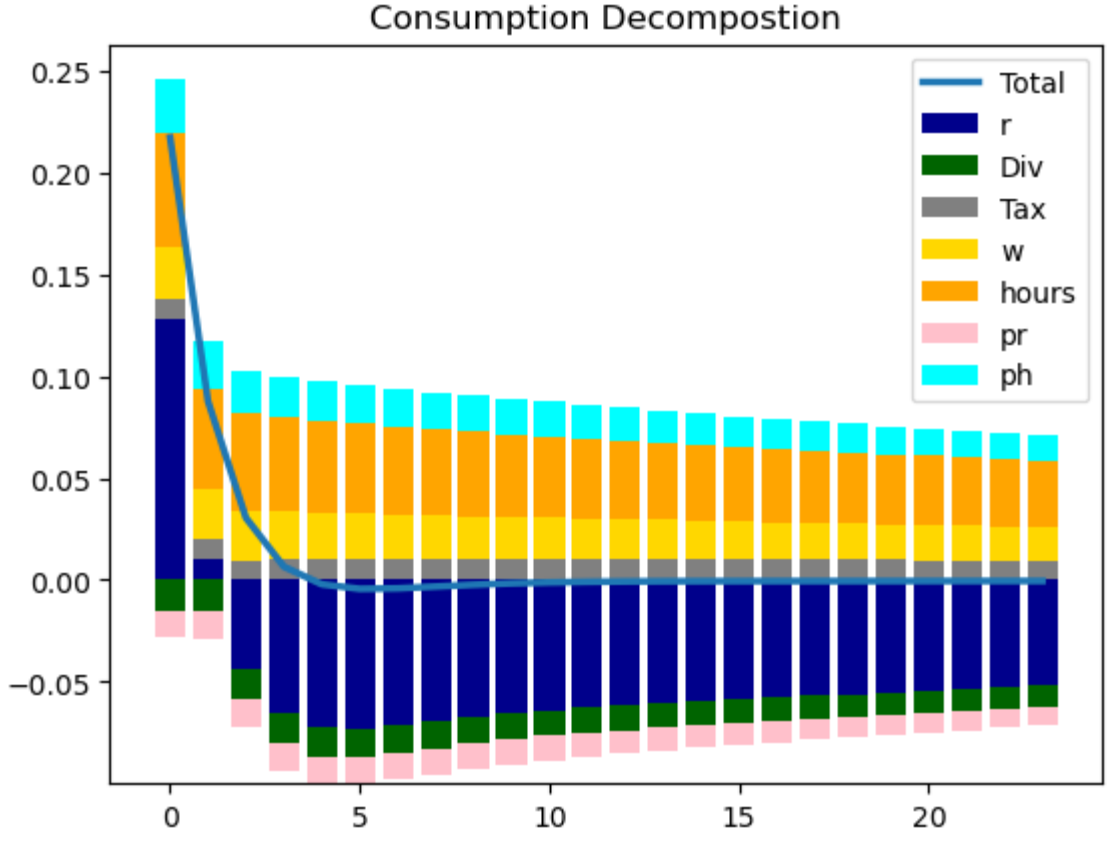


Decomposing Consumption

Decomposing Consumption

In [16]:

```
plot_C_decomp = hhprob_WD_st.plot_C_decomp # function to plot Irfs
plot_C_decomp(IPRs,J_hh,ss0,T) # base
```



Including sticky expectations

- Suppose when an aggregate shock occurs,
    - In any given period,
    - 25% of hh do not update their forecast of the path of aggregate variables.

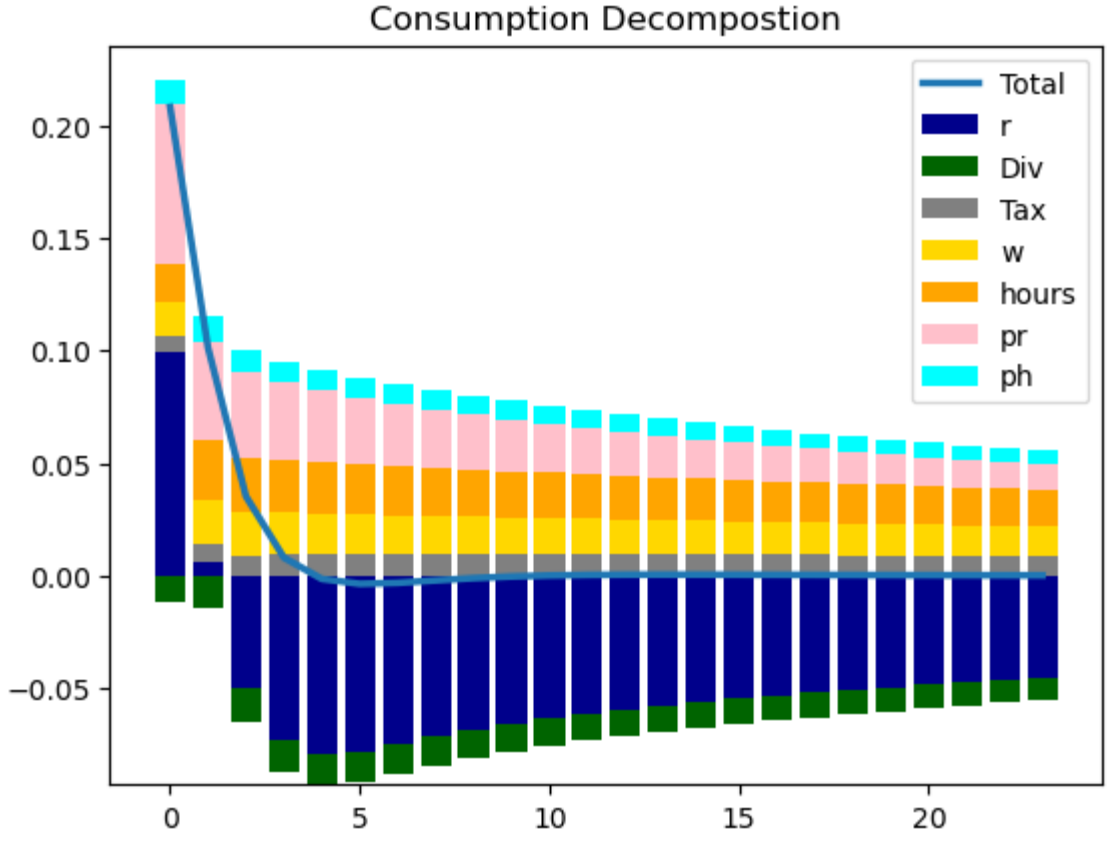
In [17]:

```
%%time
theta = .25 # Sticky expectations parameter

stick_jacob = hhprob_WD_st.stick_jacob

J_hh_sticky = stick_jacob(J_hh,theta)

G_sticky = hank.solve_jacobian(ss0, unknowns, targets, exogenous, T=T,Js={'hh': J_hh_sticky})
# convert G jacobian dictionary into impulse response dictionary
IPRs_sticky = {}
for output in G_sticky:
    IPRs_sticky[output] = G_sticky[output]['epsr'] @drstar
```

```
CPU times: user 9.52 s, sys: 3.14 s, total: 12.7 s
Wall time: 4.65 s
```

In [18]:

```
plot_C_decomp(IPRs_sticky,J_hh_sticky,ss0,T) # base
```

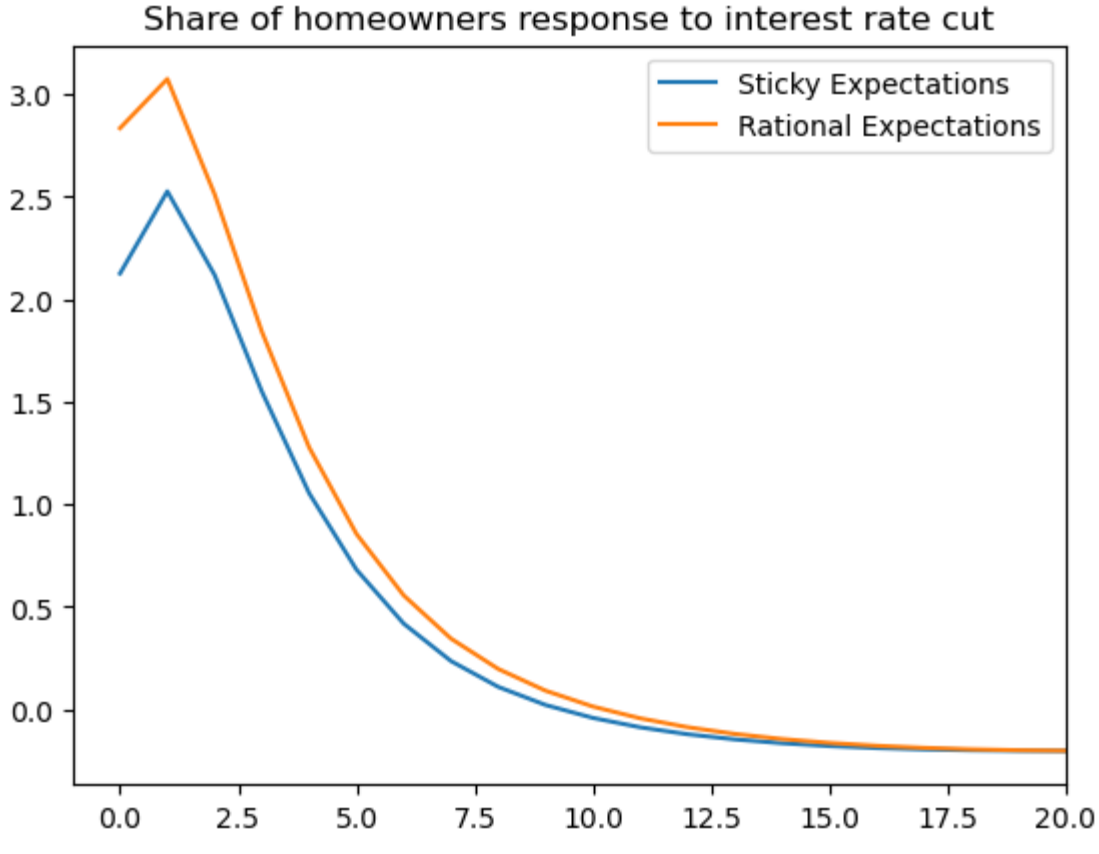


In [19]:

```
plt.plot(100 * J_hh_sticky['HOO']['r']@IPRs_sticky['r']/ss0['HOO'], label =  'Sticky Expectations')
plt.plot(100 * J_hh_base['HOO']['r']@IPRs_sticky['r']/ss0['HOO'], label = 'Rational Expectations')
plt.xlim(-1,20)
plt.title('Share of homeowners response to interest rate cut')
plt.legend()
```

Out[19]:

```
<matplotlib.legend.Legend at 0x7feac8f498e0>
```



Future directions

- Add mortgages as second asset
- Add quality of housing to utility function
- Add landlords
- Decompose consumption responses between wealth effects and collateral effects