PS3

October 4, 2021

1 Problem Set 3

[11]:
```
using Distributed, SharedArrays
cd("/home/mitchv34/Work/2nd Year/ECON 899 (Computational Methods)/1st Quarter/
 ↪Problem Sets/Shared Repo/Shared Repo")
include("./PS3/JuliaCode/conesa_kueger.jl");
```

[14]:
```
using Plots, LaTeXStrings
theme(:juno) # Comment this line for final version
# theme(:wong2) # un-coment for final version
default(fontfamily="Computer Modern", framestyle=:box) # LaTex-style
```

1.1 Exercise 1

Solve the dynamic programming problem of retirees and workers. Plot the value function over a for a retired agent at the model-age 50 . Is it increasing and concave? Plot the savings function for a worker at the model-age 20, $a'_{20}(z, a)$. Is saving increasing in a? Is it increasing in z?

[12]:
```
# Initialize the model

prim, res = Initialize();
```

Here we include some interesting stuff like

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

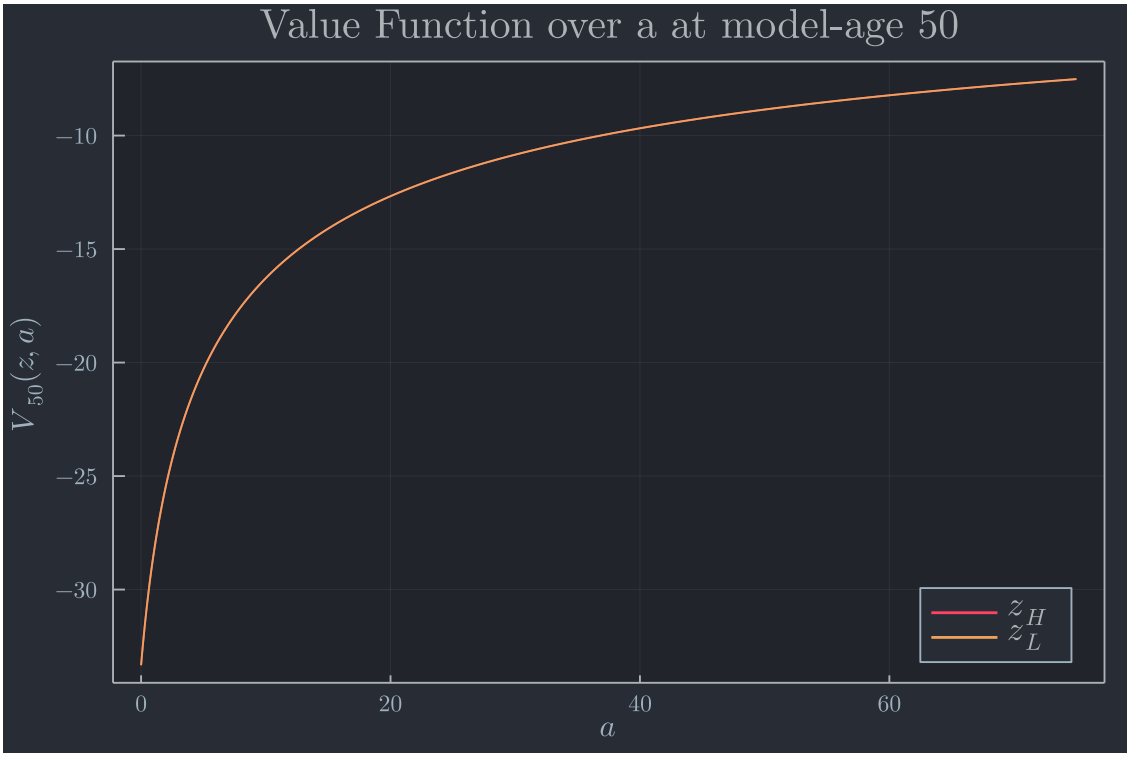
[13]:
```
# Solving the value function for a predetermined values of r, w, and b
@time begin
        V_ret(prim, res)
        V_workers(prim, res)
end

# Jupyter dosen't like progress bars
```

```
  2.381606 seconds (636.92 k allocations: 342.476 MiB, 1.94% gc time, 16.95%
compilation time)
```

[15]:
```
plot( prim.a_grid, res.val_fun[:,1,50], label = L"z_H", legend = :bottomright,␣
↪legendfontsize=12)
plot!(prim.a_grid ,res.val_fun[:,2,50], label = L"z_L")
xlabel!(L"a")
ylabel!(L"V_{50}(z,a)")
title!("Value Function over a at model-age 50")
```

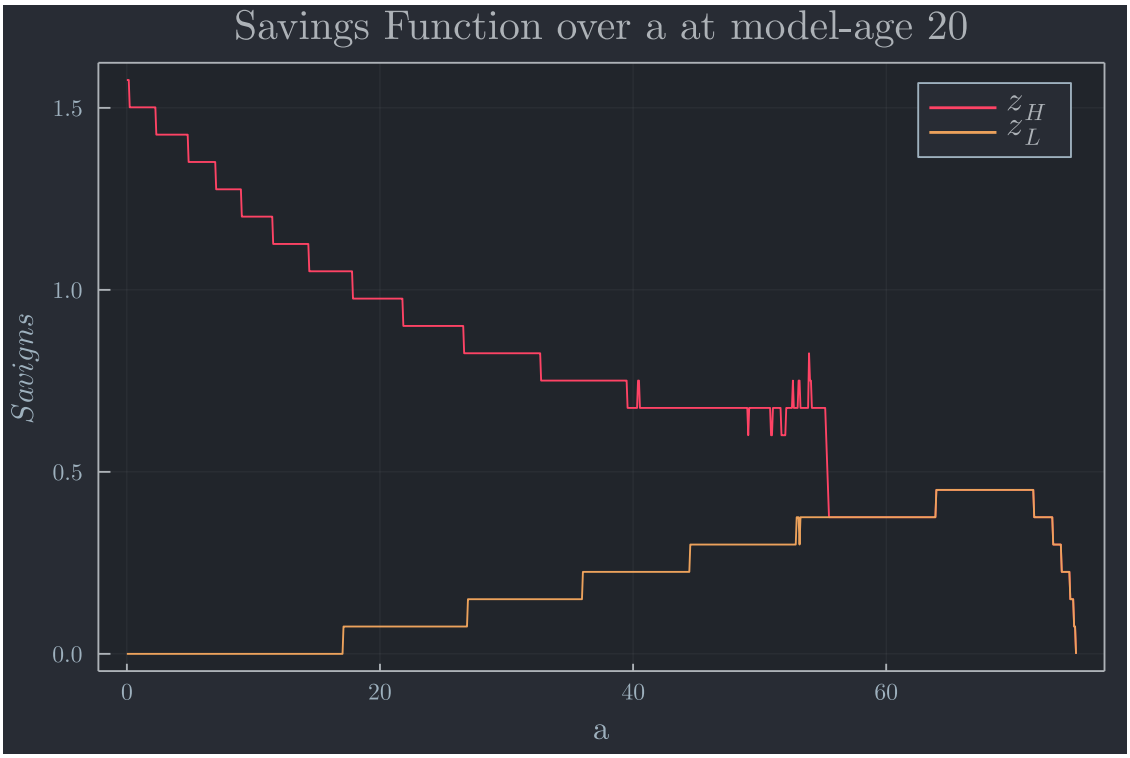
[15]:



[16]:
```
# Calculate savings
savings = res.pol_fun[:,:,:] .- prim.a_grid;
```

[17]:
```
plot( prim.a_grid, savings[:,1,20], label = L"z_H",  legendfontsize=12)
plot!(prim.a_grid ,savings[:,2,20], label = L"z_L")
xlabel!("a")
ylabel!(L"Savigns")
title!("Savings Function over a at model-age 20")
```

[17]:



1.2 Exercise 2

Exercise 2 After solving for agent's dynamic programming problem, solve for the steady-state distribution of agents over age, productivity and asset holdings, $F_j(z, a)$. Find first the relative sizes of each cohort of age j (denoted by μ_j) using the expression below:

$$\mu_{i+1} = \frac{\mu_i}{1+n}, \text{ for } i = 1, \ldots, N-1$$

with any $\mu_1 = \tilde{\mu}_1 > 0$. Then normalize μ, so that it sums up to 1 across all age groups. Finally, start with the newborn generation with zero wealth: given its distribution, $F_1\left(z^H, 0\right) = \mu_1 \times 0.2037$ and $F_1\left(z^L, 0\right) = \mu_1 \times 0.7963$, compute the distribution of agents over asset holdings at subsequent ages by applying the optimal decision rules.

Here we solve and show that we solved the question.

1.3 Exercise 3

We have computed the decision rules and the stationary distribution for given prices. There is no guarantee that at these prices the supply of assets and labor by the agents equals the demand for capital and labor by the firms. In order to find the equilibrium prices, we use the "guess and verify" method. First, we make initial guesses on aggregate capital and aggregate labor, demanded by the firm, which we denote K^0 and L^0, respectively. They imply the interest rate r^0 and wage w^0, since markets are perfectly competitive. Observe that from the guess on K and L we can compute the

pension benefit, b, using the government bud get constraint:

$$b = \frac{\theta w^0 L^0}{\sum_{j=J^R}^{N} \mu_j}$$

Given r^0 and w^0, we compute the optimal decision rules and the stationary distribution (you have already done that!). Finally, we verify, if our guess was correct by computing the aggregate assets and labor supplied by households:

$$K^{new} = \sum_{j=1}^{N} \sum_{m=1}^{n_a} \sum_{z \in \{z^L, z^H\}} F_j\left(z, a_m\right) a_m \text{ and } L^{new} = \sum_{j=1}^{J^R-1} \sum_{m=1}^{n_a} \sum_{z \in \{z^L, z^H\}} F_j\left(z, a_m\right) e\left(z, \eta_j\right) l_j\left(z, a_m\right)$$

If the guess was "far off" the obtained values, we update our initial guess with $K^1 = 0.99K^0 + 0.01K^{\text{new}}$ and $L^1 = 0.99L^0 + 0.01L^{\text{new}}$ and repeat the procedure. We proceed so until the guess and the updated values for K and L are "sufficiently close"

[18]:
```
@time out_prim, out_res = MarketClearing(use_Fortran=false, tol = 1e-3);
```

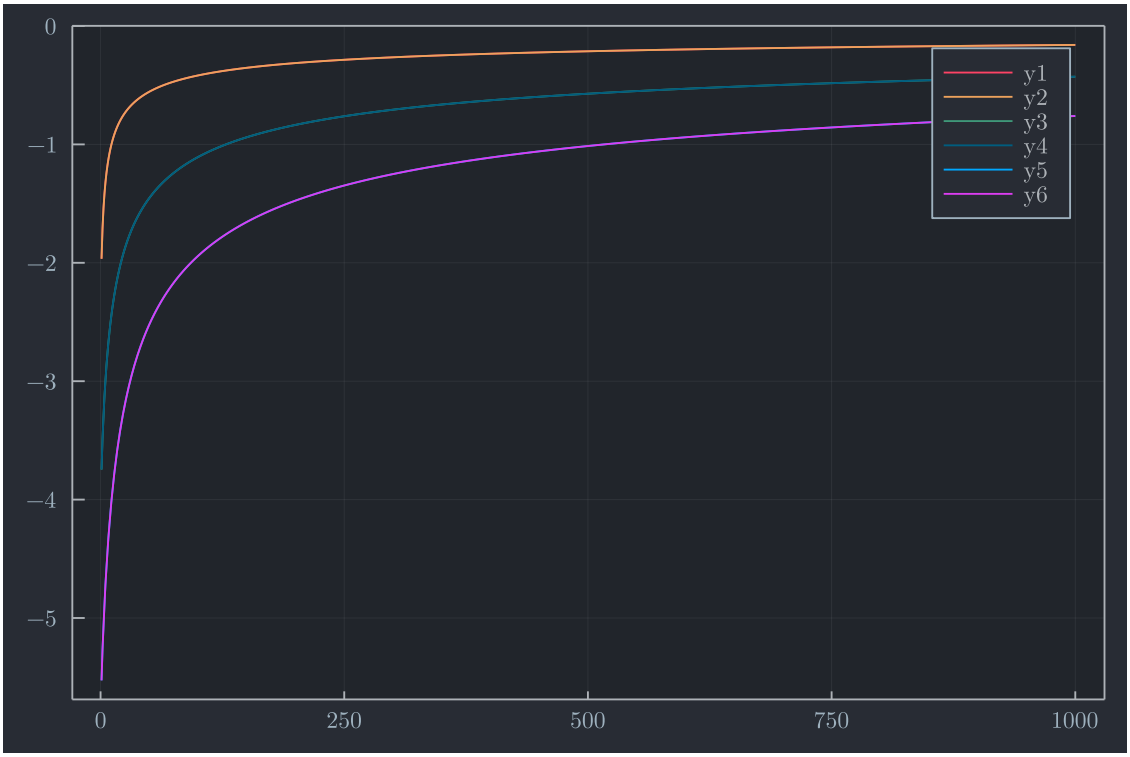
```
1 iterations; err = 4.352273967659059, K = 5.3057, L = 0.7196,   = 0.7
2 iterations; err = 0.41776053874646735, K = 5.1804, L = 0.6063,   = 0.7
3 iterations; err = 0.6496105991250047, K = 4.9855, L = 0.5291,   = 0.7
4 iterations; err = 0.6106739965762449, K = 4.8023, L = 0.4764,   = 0.7
5 iterations; err = 0.5242179867118528, K = 4.645, L = 0.4403,   = 0.7
6 iterations; err = 0.40863005245480544, K = 4.5224, L = 0.4156,   = 0.7
7 iterations; err = 0.30335235711947917, K = 4.4314, L = 0.3989,   = 0.7
8 iterations; err = 0.2206741766333753, K = 4.3652, L = 0.3874,   = 0.7
9 iterations; err = 0.1497562499196814, K = 4.3203, L = 0.3796,   = 0.7
10 iterations; err = 0.10677668550939678, K = 4.2882, L = 0.3743,   = 0.7
11 iterations; err = 0.0687575867575223, K = 4.2676, L = 0.3707,   = 0.7
12 iterations; err = 0.049407678842873004, K = 4.2528, L = 0.3682,   = 0.7
13 iterations; err = 0.03363659246425854, K = 4.2427, L = 0.3665,   = 0.7
14 iterations; err = 0.026249886475333106, K = 4.2348, L = 0.3653,   = 0.7
15 iterations; err = 0.018944001520645237, K = 4.2291, L = 0.3645,   = 0.7
16 iterations; err = 0.013199742153609861, K = 4.2252, L = 0.364,   = 0.7
17 iterations; err = 0.009073670278540114, K = 4.2238, L = 0.3638,   = 0.85
18 iterations; err = 0.007754571418216116, K = 4.2227, L = 0.3637,   = 0.85
19 iterations; err = 0.006484705735621432, K = 4.2217, L = 0.3635,   = 0.85
20 iterations; err = 0.005339974526787472, K = 4.2209, L = 0.3634,   = 0.85
21 iterations; err = 0.004572481105361348, K = 4.2204, L = 0.3633,   = 0.9
22 iterations; err = 0.003887340475794865, K = 4.22, L = 0.3633,   = 0.9
23 iterations; err = 0.003529963352272958, K = 4.2197, L = 0.3632,   = 0.9
24 iterations; err = 0.0032267097454461435, K = 4.2194, L = 0.3632,   = 0.9
25 iterations; err = 0.002882400557919773, K = 4.2191, L = 0.3632,   = 0.9
26 iterations; err = 0.0026700527596661416, K = 4.2188, L = 0.3631,   = 0.9
27 iterations; err = 0.0024111410191043348, K = 4.2186, L = 0.3631,   = 0.9
28 iterations; err = 0.0021857428055058747, K = 4.2184, L = 0.3631,   = 0.9
29 iterations; err = 0.0019159666024313182, K = 4.2182, L = 0.363,   = 0.9
30 iterations; err = 0.0017164066707104908, K = 4.218, L = 0.363,   = 0.9
```

```
31 iterations; err = 0.00153617176554377, K = 4.2178, L = 0.363,   = 0.9
32 iterations; err = 0.0014798211287461172, K = 4.2177, L = 0.363,   = 0.9
33 iterations; err = 0.001328057357905088, K = 4.2176, L = 0.363,   = 0.9
34 iterations; err = 0.001152092150538131, K = 4.2174, L = 0.3629,   = 0.9
35 iterations; err = 0.0010383672751590467, K = 4.2173, L = 0.3629,   = 0.9
36 iterations; err = 0.0009667613950350429, K = 4.2173, L = 0.3629,   = 0.95
 90.176492 seconds (69.28 M allocations: 47.768 GiB, 3.68% gc time, 0.01%
compilation time)
```

Next we plot all the figures:

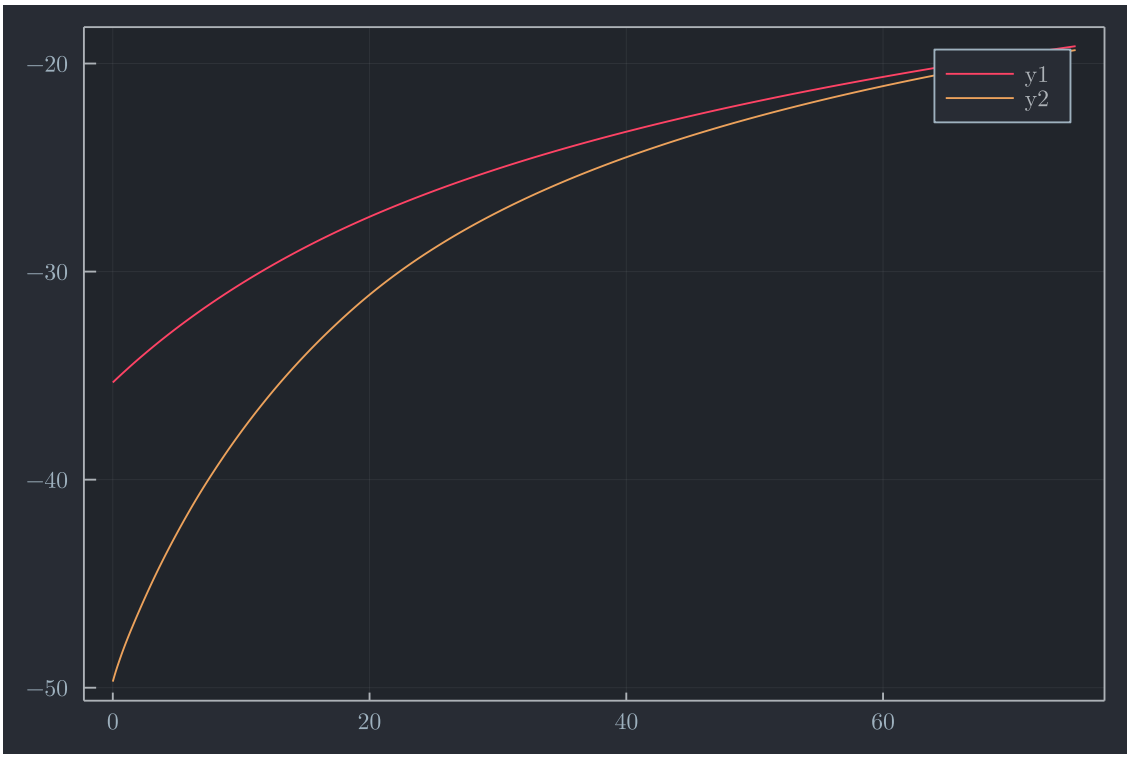
[19]:
```
plot(out_res.val_fun[:,:, end])
plot!(out_res.val_fun[:,:, end-1])
plot!(out_res.val_fun[:,:, end-2])
```

[19]:



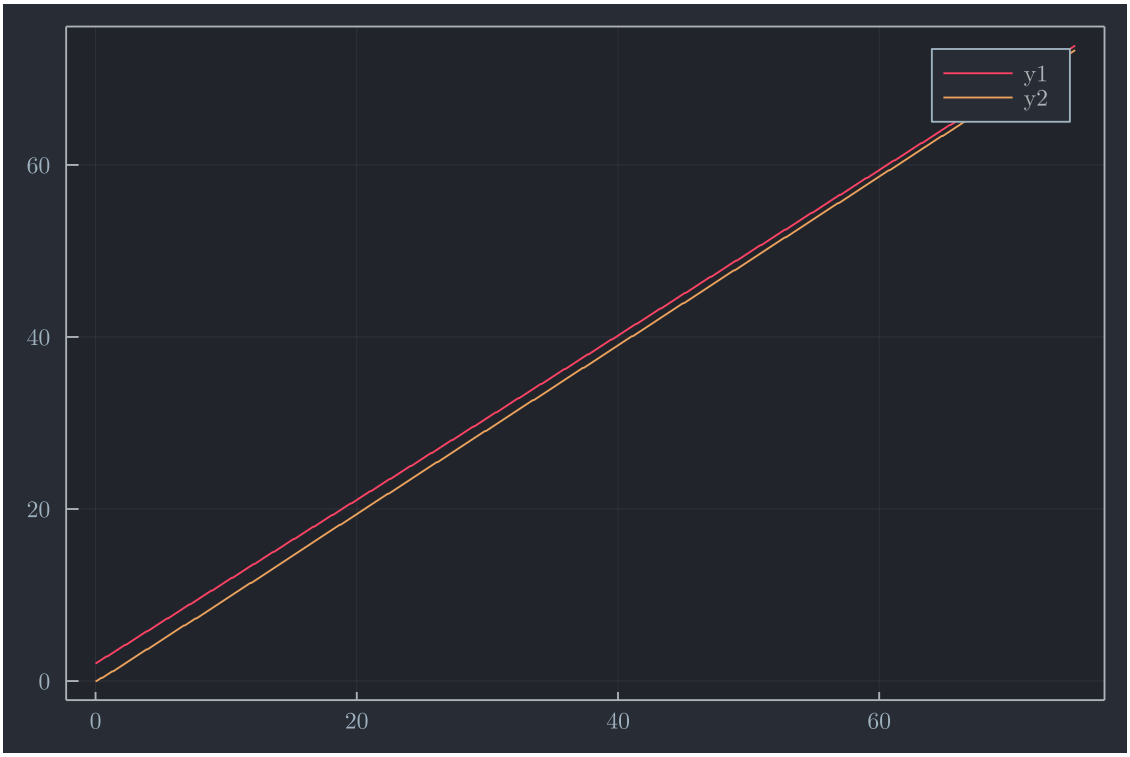
[20]:
```
plot(out_prim.a_grid, out_res.val_fun[:, 1, end])
plot!(out_prim.a_grid, out_res.val_fun[:, 2, end])
plot(out_prim.a_grid, out_res.val_fun[:, 1, 20])
plot!(out_prim.a_grid, out_res.val_fun[:, 2,20])
```

[20]:



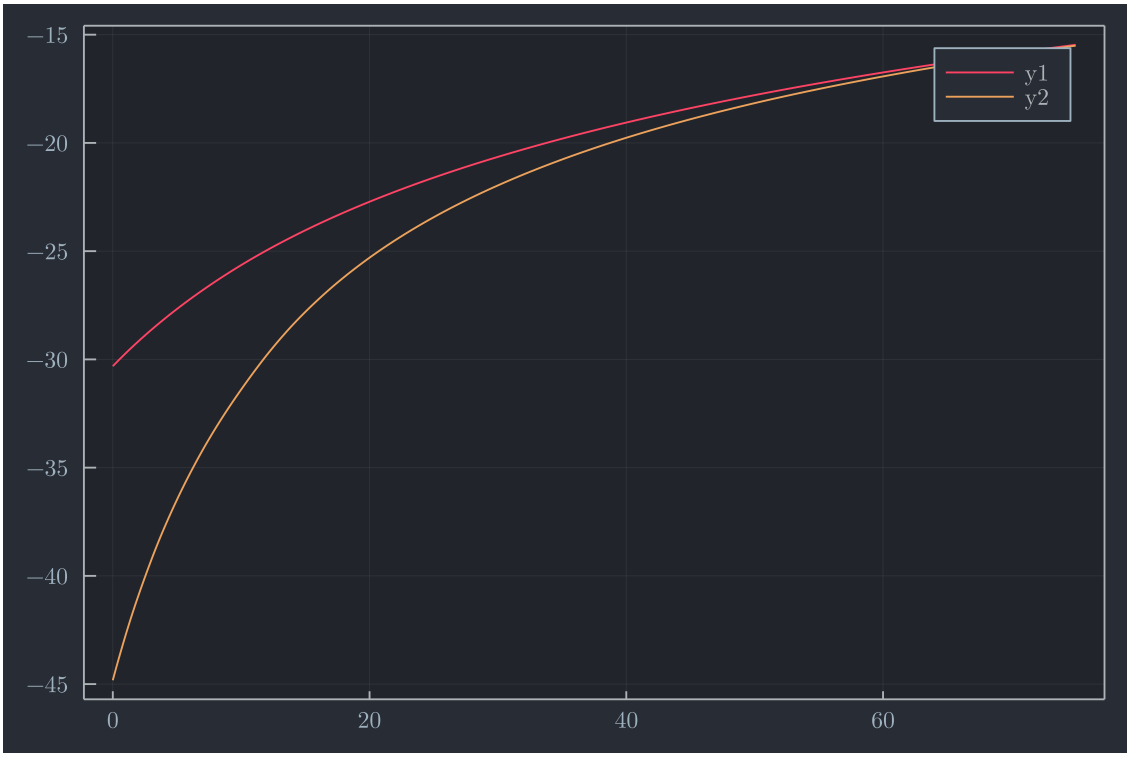
[21]:
```
plot(out_prim.a_grid, out_res.pol_fun[:,1,20])
plot!(out_prim.a_grid,out_res.pol_fun[:,2,20])
```

[21]:



[22]:
```
plot(out_prim.a_grid, out_res.val_fun[:, 1, 34])
plot!(out_prim.a_grid, out_res.val_fun[:, 2, 34])
```

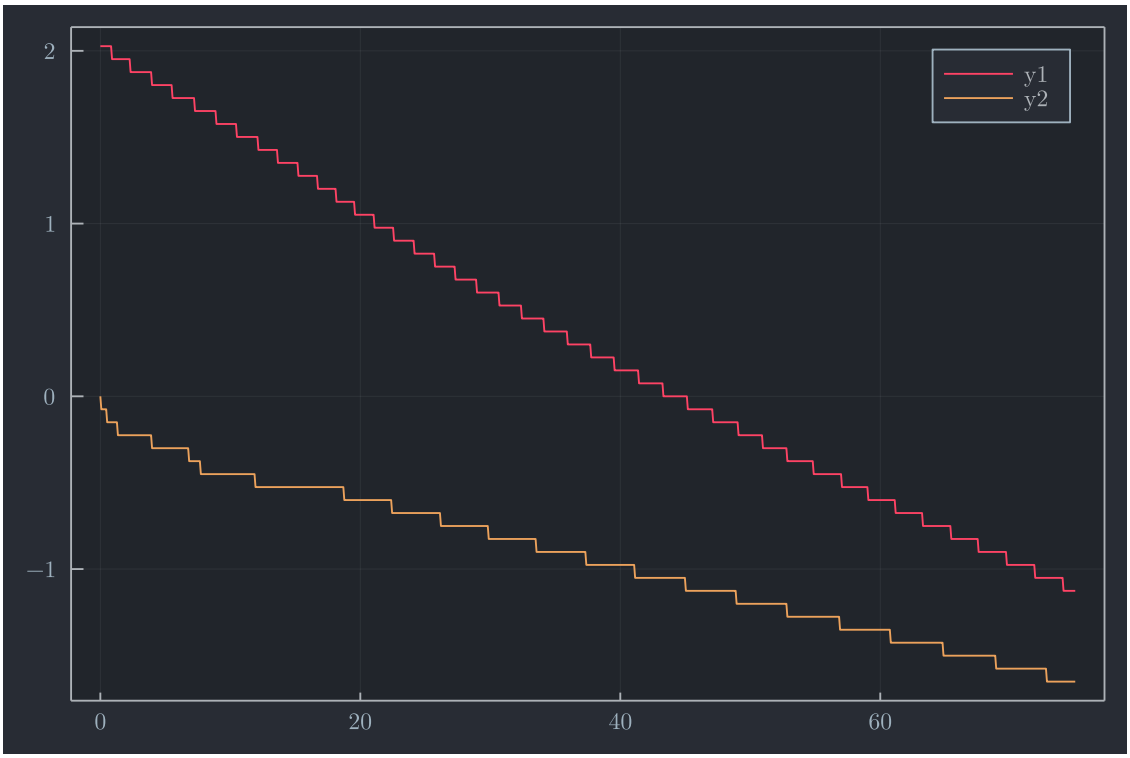
[22]:



[23]:
```
# Calculate savings
savings = out_res.pol_fun[:,:,:] .- out_prim.a_grid;
```

[24]:
```
plot(out_prim.a_grid, savings[:,1,20])
plot!(out_prim.a_grid, savings[:,2,20])
```

[24]:



[25]:
```
#Plotting Policy Functions
    #Find a_hat (The asset point beyond which everyone dissaves)
    function Find_a_hat()
        a_hat=zeros(out_prim.nZ, out_prim.N_final)
        for zi=1:out_prim.nZ, ni=1:out_prim.N_final
            for ai=1:out_prim.nA
                if out_res.pol_fun[ai,zi,ni]<=out_prim.a_grid[ai]
                    a_hat[zi,ni]=out_prim.a_grid[ai]
                    break
                end
            end
        end
        return a_hat
    end
```

[25]:
```
Find_a_hat (generic function with 1 method)
```

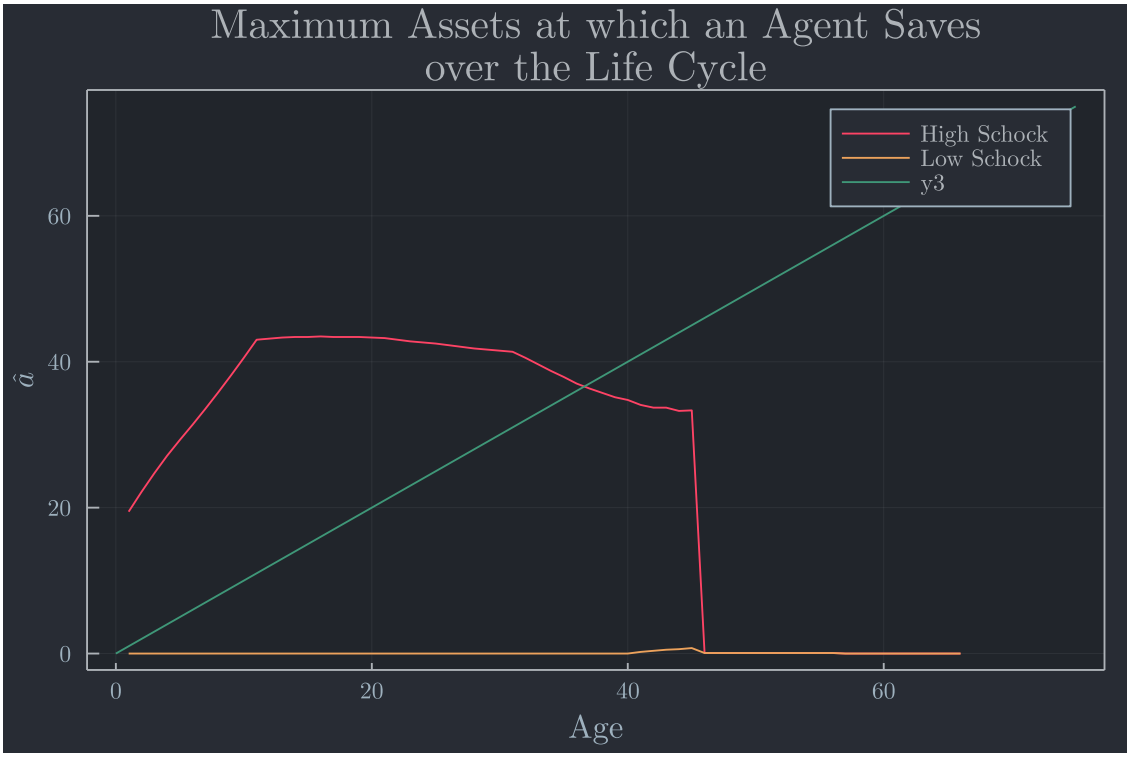
[36]:
```
a_hat=Find_a_hat()
    plot(out_prim.a_grid, out_res.pol_fun[:,1,20])
    plot!(out_prim.a_grid, out_res.pol_fun[:,2,20])
    vline!([a_hat[1,20]], label=L"\hat{a}")

    plot(1:out_prim.N_final,a_hat[1,:],
```

```
        xlabel="Age", ylabel=L"\hat{a}",label="High Schock")
    plot!(1:out_prim.N_final,a_hat[2,:], title="Maximum Assets at which an␣
 ↪Agent Saves
over the Life Cycle",
        xlabel="Age", ylabel=L"\hat{a}",label="Low Schock")

plot!(out_prim.a_grid, out_prim.a_grid)
```

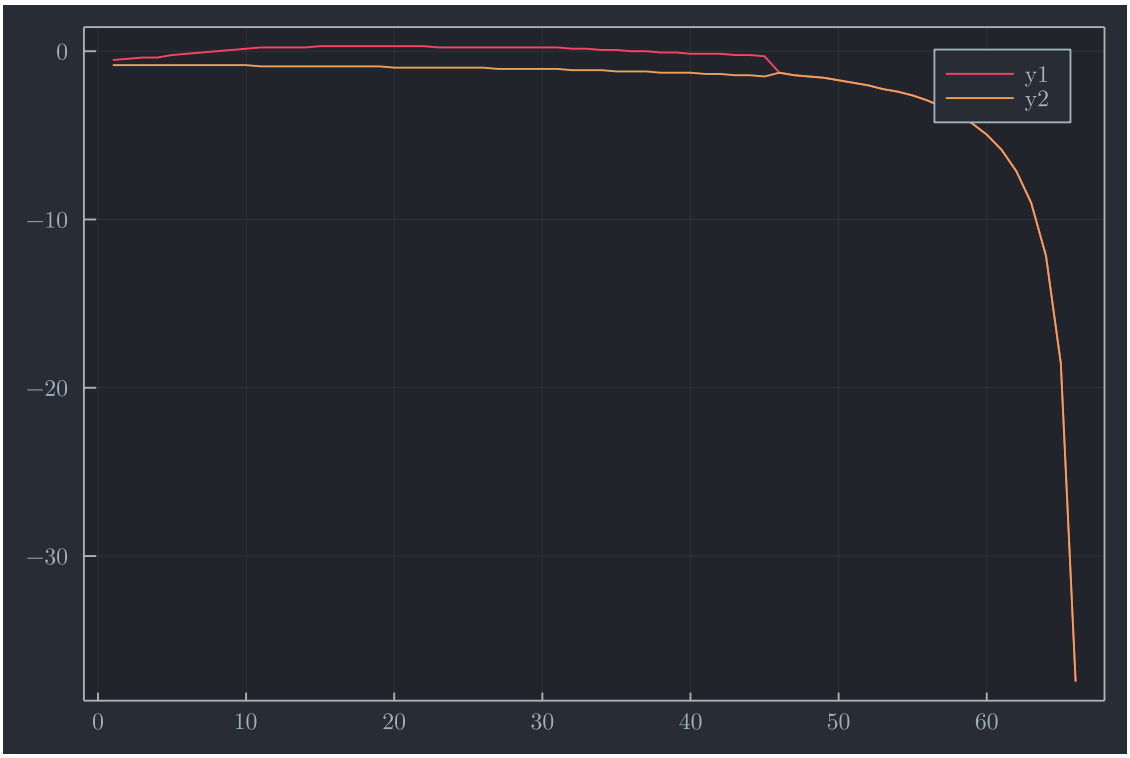
[36]:



[37]:

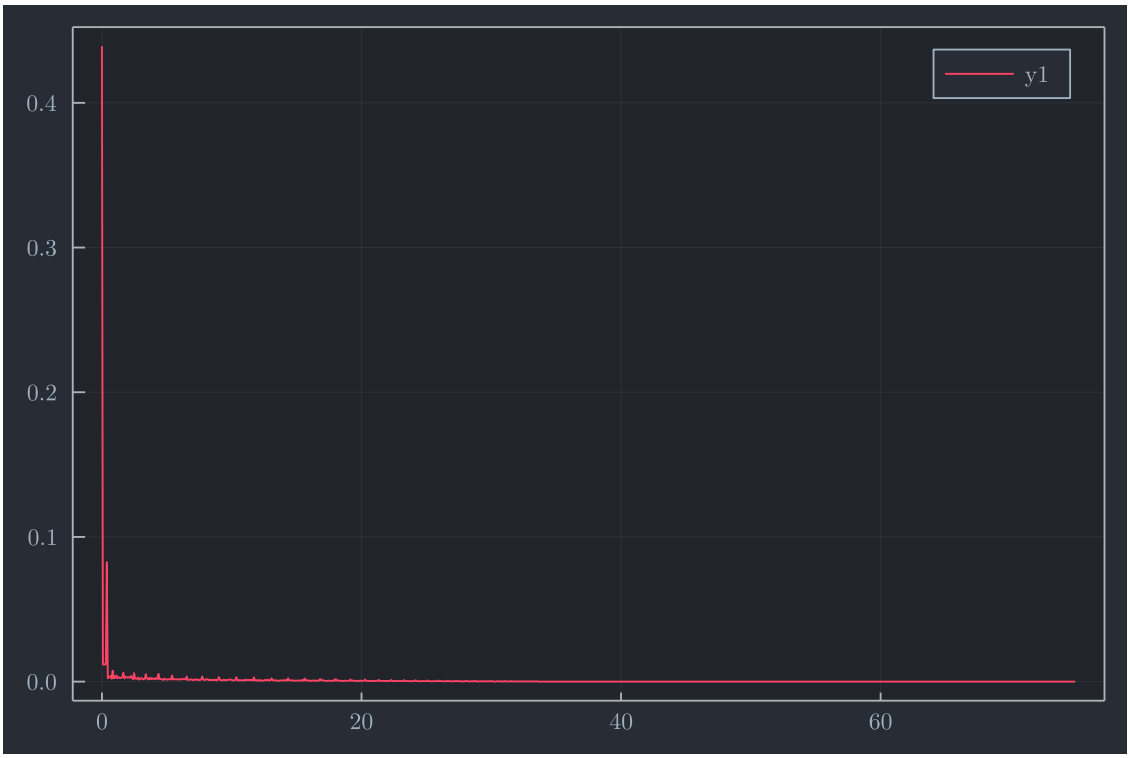
```
plot(savings[500,1,:])
plot!(savings[500,2,:])
```

[37]:



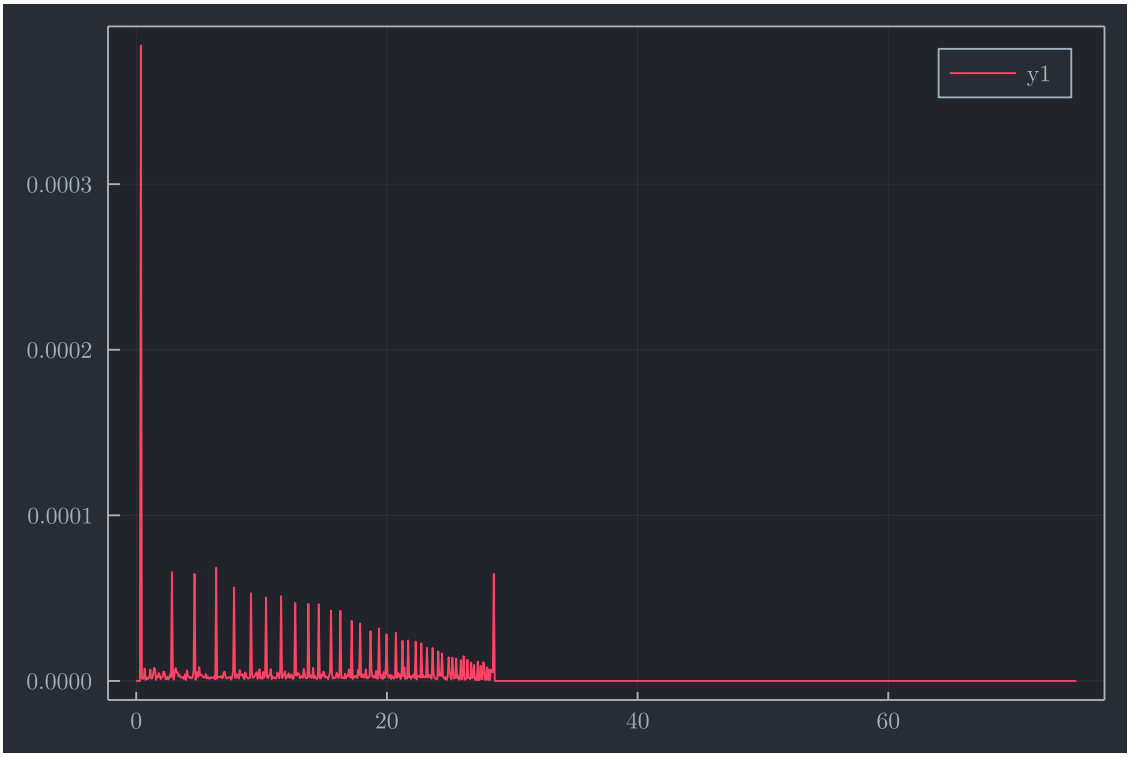
[39]:
```
# accross assets for workers and retirees
a_dist = sum(out_res.F, dims = 2:3)
plot(out_prim.a_grid, a_dist[:, 1, 1])
```

[39]:



[40]:
```
plot(out_prim.a_grid,out_res.F[:,1,50])
```

[40]:



You are now asked to evaluate the macroeconomic consequences of eliminating social security. 1. First, solve for the benchmark model with social security. Is this economy dynamically efficient (compare the interest rate with the implicit return from social security, which is equal to the population growth rate)? Now eliminate social security by setting $\theta = 0$. Observe how aggregate capital accumulation and labor supply change as a result of the tax reform. Provide intuition in terms of insurance and output efficiency. How does aggregate welfare change? Who benefits and who loses due to this reform? How does the reform affect cross-sectional wealth inequality? You can use table 1 to support your answers. 2. In the second experiment, there is no idiosyncratic risk. Assume that at each age j, $z^L = z^H = 0.5$. First, compute the aggregate variables for the case with social security. How does the aggregate capital stock change relative to the benchmark model? Provide intuition in terms of capital as a buffer stock. Then, eliminate social security. How does the aggregate welfare change? What can you conclude about social security as an insurance device against idiosyncratic risk? Comment on the extent, to which these welfare comparisons across steady states are meaningful or misleading. 3. Consider the case, when labor supply is exogenous ($\gamma = 1$). Compare the distortionary effect of social security on the aggregate labor supply. How does the support for social security change with exogenous labor supply?

[41]:
```
# conduct policy experiments
@time prim_noSS, res_noSS  = MarketClearing(use_Fortran=false, tol = 1e-3, ss =␣
 ↪false);
```

```
1 iterations; err = 5.580099898018354, K = 5.674, L = 0.7211,   = 0.7
2 iterations; err = 0.3684682726372762, K = 5.596, L = 0.6105,   = 0.7
3 iterations; err = 0.5401578446475943, K = 5.434, L = 0.5354,   = 0.7
4 iterations; err = 0.5795025981134385, K = 5.2601, L = 0.4842,   = 0.7
5 iterations; err = 0.469260726824702, K = 5.1193, L = 0.4492,   = 0.7
6 iterations; err = 0.36958185305619473, K = 5.0084, L = 0.4255,   = 0.7
7 iterations; err = 0.2626134693420754, K = 4.9297, L = 0.4093,   = 0.7
8 iterations; err = 0.18260137262388998, K = 4.8749, L = 0.3982,   = 0.7
9 iterations; err = 0.12345221646625859, K = 4.8378, L = 0.3907,   = 0.7
10 iterations; err = 0.09250892726078686, K = 4.8101, L = 0.3856,   = 0.7
11 iterations; err = 0.06955408369980276, K = 4.7892, L = 0.3821,   = 0.7
12 iterations; err = 0.04092417091698408, K = 4.777, L = 0.3798,   = 0.7
13 iterations; err = 0.03036382603584542, K = 4.7678, L = 0.3782,   = 0.7
14 iterations; err = 0.02299090926484393, K = 4.7609, L = 0.3771,   = 0.7
15 iterations; err = 0.01757183058176981, K = 4.7557, L = 0.3764,   = 0.7
16 iterations; err = 0.011936903279424982, K = 4.7521, L = 0.3759,   = 0.7
17 iterations; err = 0.007545684647130457, K = 4.751, L = 0.3757,   = 0.85
18 iterations; err = 0.0058973324564775, K = 4.7501, L = 0.3755,   = 0.85
19 iterations; err = 0.005382725871192484, K = 4.7493, L = 0.3754,   = 0.85
20 iterations; err = 0.004550130622099857, K = 4.7488, L = 0.3753,   = 0.9
21 iterations; err = 0.004012962859793134, K = 4.7484, L = 0.3753,   = 0.9
22 iterations; err = 0.0036532154976924858, K = 4.748, L = 0.3752,   = 0.9
23 iterations; err = 0.0037464460559748503, K = 4.7477, L = 0.3752,   = 0.9
24 iterations; err = 0.0034151745710229164, K = 4.7473, L = 0.3751,   = 0.9
```

```
25 iterations; err = 0.003101151603582153, K = 4.747, L = 0.3751,   = 0.9
26 iterations; err = 0.0027821721762570917, K = 4.7467, L = 0.3751,   = 0.9
27 iterations; err = 0.002507111004000251, K = 4.7465, L = 0.375,   = 0.9
28 iterations; err = 0.0021419282270542794, K = 4.7463, L = 0.375,   = 0.9
29 iterations; err = 0.0018707684585530515, K = 4.7461, L = 0.375,   = 0.9
30 iterations; err = 0.0016835588648618227, K = 4.7459, L = 0.3749,   = 0.9
31 iterations; err = 0.001477622754576302, K = 4.7458, L = 0.3749,   = 0.9
32 iterations; err = 0.0013297288838432409, K = 4.7456, L = 0.3749,   = 0.9
33 iterations; err = 0.0012373045837952229, K = 4.7455, L = 0.3749,   = 0.9
34 iterations; err = 0.0011180530777877157, K = 4.7454, L = 0.3749,   = 0.9
35 iterations; err = 0.0006926640494375746, K = 4.7454, L = 0.3749,   = 0.95
 89.093512 seconds (67.35 M allocations: 46.441 GiB, 3.90% gc time)
```

[42]:
```
@time prim_noRisk, res_noRisk  = MarketClearing(use_Fortran=false, tol = 1e-3,␣
 ↪i_risk = false);
```

```
1 iterations; err = 1.6008716874903914, K = 3.5197, L = 0.6682,   = 0.7
2 iterations; err = 2.2075868276158515, K = 2.8575, L = 0.512,   = 0.7
3 iterations; err = 1.5607614671926942, K = 2.3892, L = 0.4041,   = 0.7
4 iterations; err = 1.1204927002535128, K = 2.0531, L = 0.3299,   = 0.7
5 iterations; err = 0.7615825965343535, K = 1.8246, L = 0.2787,   = 0.7
6 iterations; err = 0.5852954430657837, K = 1.649, L = 0.2437,   = 0.7
7 iterations; err = 0.3652887276516028, K = 1.5394, L = 0.2195,   = 0.7
8 iterations; err = 0.23936317905305682, K = 1.4676, L = 0.203,   = 0.7
9 iterations; err = 0.206842109886213, K = 1.4056, L = 0.1917,   = 0.7
10 iterations; err = 0.13159223815782117, K = 1.3661, L = 0.184,   = 0.7
11 iterations; err = 0.0921145667104748, K = 1.3385, L = 0.1787,   = 0.7
12 iterations; err = 0.06089864237777376, K = 1.3202, L = 0.1752,   = 0.7
13 iterations; err = 0.0426290496644417, K = 1.3074, L = 0.1727,   = 0.7
14 iterations; err = 0.029840334765109056, K = 1.2985, L = 0.171,   = 0.7
15 iterations; err = 0.020888234335576294, K = 1.2922, L = 0.1698,   = 0.7
16 iterations; err = 0.0062117078123360425, K = 1.2931, L = 0.1695,   = 0.85
17 iterations; err = 0.0052799516404857805, K = 1.2939, L = 0.1692,   = 0.85
18 iterations; err = 0.01634551295282649, K = 1.2915, L = 0.1689,   = 0.85
19 iterations; err = 0.006939785837336787, K = 1.2925, L = 0.1687,   = 0.85
20 iterations; err = 0.01493465388550308, K = 1.2903, L = 0.1685,   = 0.85
21 iterations; err = 0.008139016044561798, K = 1.2915, L = 0.1684,   = 0.85
22 iterations; err = 0.01391530820936171, K = 1.2894, L = 0.1682,   = 0.85
23 iterations; err = 0.009005459869281918, K = 1.2907, L = 0.1681,   = 0.85
24 iterations; err = 0.013178830958349819, K = 1.2888, L = 0.168,   = 0.85
25 iterations; err = 0.009631465532641936, K = 1.2902, L = 0.168,   = 0.85
26 iterations; err = 0.012646726144493803, K = 1.2883, L = 0.1679,   = 0.85
27 iterations; err = 0.010083754624419683, K = 1.2898, L = 0.1679,   = 0.85
28 iterations; err = 0.012262280416482696, K = 1.288, L = 0.1678,   = 0.85
29 iterations; err = 0.01041053349322918, K = 1.2895, L = 0.1678,   = 0.85
30 iterations; err = 0.011984518377994524, K = 1.2878, L = 0.1677,   = 0.85
31 iterations; err = 0.010646631225944114, K = 1.2893, L = 0.1677,   = 0.85
32 iterations; err = 0.011783835305186896, K = 1.2876, L = 0.1677,   = 0.85
```

```
33 iterations; err = 0.010817211837830465, K = 1.2892, L = 0.1677,   = 0.85
34 iterations; err = 0.002506299432448955, K = 1.289, L = 0.1677,   = 0.9
35 iterations; err = 0.009445260005400913, K = 1.2899, L = 0.1677,   = 0.9
36 iterations; err = 0.003200195489744262, K = 1.2896, L = 0.1677,   = 0.9
37 iterations; err = 0.0028801759407697247, K = 1.2893, L = 0.1677,   = 0.9
38 iterations; err = 0.0025921583466925746, K = 1.289, L = 0.1677,   = 0.9
39 iterations; err = 0.009367986982581344, K = 1.29, L = 0.1677,   = 0.9
40 iterations; err = 0.0032697412102817847, K = 1.2896, L = 0.1677,   = 0.9
41 iterations; err = 0.0029427670892534508, K = 1.2893, L = 0.1677,   = 0.9
42 iterations; err = 0.011781032732962604, K = 1.2882, L = 0.1677,   = 0.9
43 iterations; err = 0.010230542387572816, K = 1.2892, L = 0.1677,   = 0.9
44 iterations; err = 0.002493441345789549, K = 1.2889, L = 0.1677,   = 0.9
45 iterations; err = 0.009456832283394112, K = 1.2899, L = 0.1677,   = 0.9
46 iterations; err = 0.003189780439550205, K = 1.2896, L = 0.1677,   = 0.9
47 iterations; err = 0.002870802395595007, K = 1.2893, L = 0.1677,   = 0.9
48 iterations; err = 0.009117207338569466, K = 1.2902, L = 0.1677,   = 0.9
49 iterations; err = 0.003495442889892564, K = 1.2898, L = 0.1677,   = 0.9
50 iterations; err = 0.0031458986009034184, K = 1.2895, L = 0.1677,   = 0.9
51 iterations; err = 0.011963851093447442, K = 1.2883, L = 0.1677,   = 0.9
52 iterations; err = 0.010066005863136551, K = 1.2893, L = 0.1677,   = 0.9
53 iterations; err = 0.0026415242177821874, K = 1.2891, L = 0.1677,   = 0.9
54 iterations; err = 0.009323557698601048, K = 1.29, L = 0.1677,   = 0.9
55 iterations; err = 0.003309727565864007, K = 1.2897, L = 0.1677,   = 0.9
56 iterations; err = 0.0029787548092774507, K = 1.2894, L = 0.1677,   = 0.9
57 iterations; err = 0.0026808793283499277, K = 1.2891, L = 0.1677,   = 0.9
58 iterations; err = 0.009288138099089904, K = 1.29, L = 0.1677,   = 0.9
59 iterations; err = 0.0033416052054240364, K = 1.2897, L = 0.1677,   = 0.9
60 iterations; err = 0.003007444684881566, K = 1.2894, L = 0.1677,   = 0.9
61 iterations; err = 0.01183924256902813, K = 1.2882, L = 0.1677,   = 0.9
62 iterations; err = 0.010178153535113976, K = 1.2892, L = 0.1677,   = 0.9
63 iterations; err = 0.002540591313002327, K = 1.289, L = 0.1677,   = 0.9
64 iterations; err = 0.009414397312902834, K = 1.2899, L = 0.1677,   = 0.9
65 iterations; err = 0.0032279719129924, K = 1.2896, L = 0.1677,   = 0.9
66 iterations; err = 0.002905174721693271, K = 1.2893, L = 0.1677,   = 0.9
67 iterations; err = 0.00908627224508085, K = 1.2902, L = 0.1677,   = 0.9
68 iterations; err = 0.0035232844740322733, K = 1.2899, L = 0.1677,   = 0.9
69 iterations; err = 0.0031709560266288683, K = 1.2896, L = 0.1677,   = 0.9
70 iterations; err = 0.011986402776600347, K = 1.2884, L = 0.1677,   = 0.9
71 iterations; err = 0.010045709348298848, K = 1.2894, L = 0.1677,   = 0.9
72 iterations; err = 0.0026597910811361203, K = 1.2891, L = 0.1677,   = 0.9
73 iterations; err = 0.009307117521582198, K = 1.29, L = 0.1677,   = 0.9
74 iterations; err = 0.0033245237251811055, K = 1.2897, L = 0.1677,   = 0.9
75 iterations; err = 0.002992071352663217, K = 1.2894, L = 0.1677,   = 0.9
76 iterations; err = 0.011825406570031394, K = 1.2882, L = 0.1677,   = 0.9
77 iterations; err = 0.010190605934211172, K = 1.2892, L = 0.1677,   = 0.9
78 iterations; err = 0.002529384153814984, K = 1.289, L = 0.1677,   = 0.9
79 iterations; err = 0.009424483756171442, K = 1.2899, L = 0.1677,   = 0.9
80 iterations; err = 0.003218894114050741, K = 1.2896, L = 0.1677,   = 0.9
```

```
81 iterations; err = 0.0028970047026457557, K = 1.2893, L = 0.1677,   = 0.9
82 iterations; err = 0.0026073042323813134, K = 1.289, L = 0.1677,   = 0.9
83 iterations; err = 0.009354355685461835, K = 1.29, L = 0.1677,   = 0.9
84 iterations; err = 0.003282009377689299, K = 1.2897, L = 0.1677,   = 0.9
85 iterations; err = 0.0029538084399205466, K = 1.2894, L = 0.1677,   = 0.9
86 iterations; err = 0.011790969948563212, K = 1.2882, L = 0.1677,   = 0.9
87 iterations; err = 0.010221598893532358, K = 1.2892, L = 0.1677,   = 0.9
88 iterations; err = 0.002501490490425917, K = 1.2889, L = 0.1677,   = 0.9
89 iterations; err = 0.009449588053221492, K = 1.2899, L = 0.1677,   = 0.9
90 iterations; err = 0.0031963002467056967, K = 1.2896, L = 0.1677,   = 0.9
91 iterations; err = 0.0028766702220353046, K = 1.2893, L = 0.1677,   = 0.9
92 iterations; err = 0.009111926294772976, K = 1.2902, L = 0.1677,   = 0.9
93 iterations; err = 0.0035001958293092716, K = 1.2898, L = 0.1677,   = 0.9
94 iterations; err = 0.0031501762463781446, K = 1.2895, L = 0.1677,   = 0.9
95 iterations; err = 0.011967700974374829, K = 1.2883, L = 0.1677,   = 0.9
96 iterations; err = 0.010062540970302125, K = 1.2893, L = 0.1677,   = 0.9
97 iterations; err = 0.0026446426213331264, K = 1.2891, L = 0.1677,   = 0.9
98 iterations; err = 0.009320751135404981, K = 1.29, L = 0.1677,   = 0.9
99 iterations; err = 0.003312253472740556, K = 1.2897, L = 0.1677,   = 0.9
100 iterations; err = 0.0029810281254665227, K = 1.2894, L = 0.1677,   = 0.9
101 iterations; err = 0.011815467665554191, K = 1.2882, L = 0.1677,   = 0.9
102 iterations; err = 0.010199550948240654, K = 1.2892, L = 0.1677,   = 0.9
103 iterations; err = 0.00252133364118845, K = 1.289, L = 0.1677,   = 0.9
104 iterations; err = 0.009431729217535212, K = 1.2899, L = 0.1677,   = 0.9
105 iterations; err = 0.0032123731988233393, K = 1.2896, L = 0.1677,   = 0.9
106 iterations; err = 0.0028911358789409647, K = 1.2893, L = 0.1677,   = 0.9
107 iterations; err = 0.0026020222910467794, K = 1.289, L = 0.1677,   = 0.9
108 iterations; err = 0.009359109432662605, K = 1.29, L = 0.1677,   = 0.9
109 iterations; err = 0.0032777310052085618, K = 1.2896, L = 0.1677,   = 0.9
110 iterations; err = 0.002949957904687661, K = 1.2894, L = 0.1677,   = 0.9
111 iterations; err = 0.011787504466853616, K = 1.2882, L = 0.1677,   = 0.9
112 iterations; err = 0.010224717827070906, K = 1.2892, L = 0.1677,   = 0.9
113 iterations; err = 0.002498683450241179, K = 1.2889, L = 0.1677,   = 0.9
114 iterations; err = 0.009452114389387978, K = 1.2899, L = 0.1677,   = 0.9
115 iterations; err = 0.0031940265441559035, K = 1.2896, L = 0.1677,   = 0.9
116 iterations; err = 0.0028746238897401355, K = 1.2893, L = 0.1677,   = 0.9
117 iterations; err = 0.009113767993838673, K = 1.2902, L = 0.1677,   = 0.9
118 iterations; err = 0.0034985383001502335, K = 1.2898, L = 0.1677,   = 0.9
119 iterations; err = 0.0031486844701351657, K = 1.2895, L = 0.1677,   = 0.9
120 iterations; err = 0.011966358375756325, K = 1.2883, L = 0.1677,   = 0.9
121 iterations; err = 0.010063749309058734, K = 1.2893, L = 0.1677,   = 0.9
122 iterations; err = 0.00264355511645209, K = 1.2891, L = 0.1677,   = 0.9
123 iterations; err = 0.00932172988979807, K = 1.29, L = 0.1677,   = 0.9
124 iterations; err = 0.0033113725937867766, K = 1.2897, L = 0.1677,   = 0.9
125 iterations; err = 0.0029802353344079435, K = 1.2894, L = 0.1677,   = 0.9
126 iterations; err = 0.011814754153601559, K = 1.2882, L = 0.1677,   = 0.9
127 iterations; err = 0.010200193108998112, K = 1.2892, L = 0.1677,   = 0.9
128 iterations; err = 0.0025207556965067823, K = 1.289, L = 0.1677,   = 0.9
```

```
129 iterations; err = 0.009432249367748 78, K = 1.2899, L = 0.1677,   = 0.9
130 iterations; err = 0.0032119050636310043, K = 1.2896, L = 0.1677,   = 0.9
131 iterations; err = 0.0028907145572678594, K = 1.2893, L = 0.1677,   = 0.9
132 iterations; err = 0.0026016431015412067, K = 1.289, L = 0.1677,   = 0.9
133 iterations; err = 0.009359450703217975, K = 1.29, L = 0.1677,   = 0.9
134 iterations; err = 0.003277423861708817, K = 1.2896, L = 0.1677,   = 0.9
135 iterations; err = 0.0029496814755378686, K = 1.2894, L = 0.1677,   = 0.9
136 iterations; err = 0.011787255680618447, K = 1.2882, L = 0.1677,   = 0.9
137 iterations; err = 0.010224941734682913, K = 1.2892, L = 0.1677,   = 0.9
138 iterations; err = 0.0024984819333904174, K = 1.2889, L = 0.1677,   = 0.9
139 iterations; err = 0.009452295754553 53, K = 1.2899, L = 0.1677,   = 0.9
140 iterations; err = 0.003193863315506862, K = 1.2896, L = 0.1677,   = 0.9
141 iterations; err = 0.0028744769839561535, K = 1.2893, L = 0.1677,   = 0.9
142 iterations; err = 0.009113900209044123, K = 1.2902, L = 0.1677,   = 0.9
143 iterations; err = 0.003498419306465328, K = 1.2898, L = 0.1677,   = 0.9
144 iterations; err = 0.0031485773758186397, K = 1.2895, L = 0.1677,   = 0.9
```

```
InterruptException:

Stacktrace:
 [1] SteadyStateDist(prim::Primitives, res::Results)
   @ Main ~/Work/2nd Year/ECON 899 (Computational Methods)/1st Quarter/Problem␣
↪Sets/Shared Repo/Shared Repo/PS3/JuliaCode/conesa_kueger.jl:273
 [2] MarketClearing(; ss::Bool, i_risk::Bool, exog_l::Bool, use_Fortran::Bool,   :
↪Float64, tol::Float64, err::Float64)
   @ Main ~/Work/2nd Year/ECON 899 (Computational Methods)/1st Quarter/Problem␣
↪Sets/Shared Repo/Shared Repo/PS3/JuliaCode/conesa_kueger.jl:321
 [3] top-level scope
   @ ./timing.jl:210 [inlined]
 [4] top-level scope
   @ ./In[42]:0
 [5] eval
   @ ./boot.jl:360 [inlined]
 [6] include_string(mapexpr::typeof(REPL.softscope), mod::Module, code::String,
↪filename::String)
   @ Base ./loading.jl:1116
```

The avove cell never converged.

[ ]:
```
@time prim_noRisk_noSS, res_noRisk_noSS = MarketClearing(use_Fortran=false, tol␣
↪= 1e-3, ss = false, i_risk = false);
```

```
1 iterations; err = 1.4074154280596076, K = 3.5778, L = 0.668,   = 0.7
2 iterations; err = 1.7427413525889714, K = 3.055, L = 0.5115,   = 0.7
3 iterations; err = 1.7785889239304677, K = 2.5214, L = 0.405,   = 0.7
4 iterations; err = 1.0414188520808598, K = 2.209, L = 0.3314,   = 0.7
5 iterations; err = 0.7823031471122919, K = 1.9743, L = 0.2811,   = 0.7
```

```
6 iterations; err = 0.548056411825544, K = 1.8098, L = 0.2464,   = 0.7
7 iterations; err = 0.40402005761080306, K = 1.6886, L = 0.2228,   = 0.7
8 iterations; err = 0.27482902733458414, K = 1.6062, L = 0.2066,   = 0.7
9 iterations; err = 0.19976477583633012, K = 1.5463, L = 0.1955,   = 0.7
10 iterations; err = 0.15792906342400848, K = 1.4989, L = 0.1881,   = 0.7
11 iterations; err = 0.1473743405961776, K = 1.4547, L = 0.1831,   = 0.7
12 iterations; err = 0.05778570608955369, K = 1.4373, L = 0.1796,   = 0.7
13 iterations; err = 0.0975639720610848, K = 1.4081, L = 0.1772,   = 0.7
14 iterations; err = 0.011180802644362053, K = 1.4047, L = 0.1754,   = 0.7
15 iterations; err = 0.045642928059222765, K = 1.391, L = 0.1744,   = 0.7
16 iterations; err = 0.0058663165667134365, K = 1.3919, L = 0.174,   = 0.85
17 iterations; err = 0.03282999712646317, K = 1.387, L = 0.1737,   = 0.85
18 iterations; err = 0.02790549755749372, K = 1.3828, L = 0.1734,   = 0.85
19 iterations; err = 0.014096693284299633, K = 1.3849, L = 0.1731,   = 0.85
20 iterations; err = 0.025834176916514906, K = 1.381, L = 0.1729,   = 0.85
21 iterations; err = 0.02195905037903767, K = 1.3777, L = 0.1728,   = 0.85
22 iterations; err = 0.019151173385987486, K = 1.3806, L = 0.1726,   = 0.85
23 iterations; err = 0.021537868830080242, K = 1.3774, L = 0.1725,   = 0.85
24 iterations; err = 0.018307188505568117, K = 1.3746, L = 0.1724,   = 0.85
25 iterations; err = 0.0407727712873589, K = 1.3807, L = 0.1723,   = 0.85
26 iterations; err = 0.021677025922836712, K = 1.3775, L = 0.1723,   = 0.85
27 iterations; err = 0.01842547203441125, K = 1.3747, L = 0.1722,   = 0.85
28 iterations; err = 0.04067223028784217, K = 1.3808, L = 0.1722,   = 0.85
29 iterations; err = 0.0217624857724259, K = 1.3776, L = 0.1722,   = 0.85
30 iterations; err = 0.018498112906562003, K = 1.3748, L = 0.1721,   = 0.85
31 iterations; err = 0.015723395970577814, K = 1.3724, L = 0.1721,   = 0.85
32 iterations; err = 0.0429689949421006, K = 1.3789, L = 0.1721,   = 0.85
33 iterations; err = 0.019810235816306143, K = 1.3759, L = 0.1721,   = 0.85
34 iterations; err = 0.016838700443860155, K = 1.3734, L = 0.1721,   = 0.85
35 iterations; err = 0.042020986139810645, K = 1.3797, L = 0.172,   = 0.85
36 iterations; err = 0.02061604329825273, K = 1.3766, L = 0.172,   = 0.85
37 iterations; err = 0.017523636803514764, K = 1.374, L = 0.172,   = 0.85
38 iterations; err = 0.014895091282987494, K = 1.3717, L = 0.172,   = 0.85
39 iterations; err = 0.04367305392655241, K = 1.3783, L = 0.172,   = 0.85
40 iterations; err = 0.019211785679522198, K = 1.3754, L = 0.172,   = 0.85
41 iterations; err = 0.0163300178275938, K = 1.3729, L = 0.172,   = 0.85
42 iterations; err = 0.04245336636363706, K = 1.3793, L = 0.172,   = 0.85
43 iterations; err = 0.020248520108000267, K = 1.3763, L = 0.172,   = 0.85
44 iterations; err = 0.01721124209180025, K = 1.3737, L = 0.172,   = 0.85
45 iterations; err = 0.014629555778030268, K = 1.3715, L = 0.172,   = 0.85
46 iterations; err = 0.043898759105766016, K = 1.3781, L = 0.172,   = 0.85
47 iterations; err = 0.019019936277190563, K = 1.3752, L = 0.172,   = 0.85
48 iterations; err = 0.016166945835611957, K = 1.3728, L = 0.172,   = 0.85
49 iterations; err = 0.042591977556821536, K = 1.3792, L = 0.172,   = 0.85
50 iterations; err = 0.020130700593793405, K = 1.3762, L = 0.172,   = 0.85
51 iterations; err = 0.01711109550472445, K = 1.3736, L = 0.172,   = 0.85
52 iterations; err = 0.014544431179015715, K = 1.3714, L = 0.172,   = 0.85
53 iterations; err = 0.04397111501492845, K = 1.378, L = 0.172,   = 0.85
```

```
54 iterations; err = 0.018958433754402426, K = 1.3752, L = 0.172,   = 0.85
55 iterations; err = 0.016114668691242118, K = 1.3728, L = 0.172,   = 0.85
56 iterations; err = 0.04263641312953581, K = 1.3792, L = 0.172,   = 0.85
57 iterations; err = 0.020092930356986338, K = 1.3761, L = 0.172,   = 0.85
58 iterations; err = 0.017078990803438332, K = 1.3736, L = 0.172,   = 0.85
59 iterations; err = 0.01451714218292266, K = 1.3714, L = 0.172,   = 0.85
60 iterations; err = 0.043994310661607416, K = 1.378, L = 0.172,   = 0.85
61 iterations; err = 0.018938717454725307, K = 1.3752, L = 0.172,   = 0.85
62 iterations; err = 0.0160979098365166, K = 1.3727, L = 0.172,   = 0.85
63 iterations; err = 0.04265065815605262, K = 1.3791, L = 0.172,   = 0.85
64 iterations; err = 0.02008082208444706, K = 1.3761, L = 0.172,   = 0.85
65 iterations; err = 0.017068698771780078, K = 1.3736, L = 0.172,   = 0.85
66 iterations; err = 0.01450839395601311, K = 1.3714, L = 0.172,   = 0.85
67 iterations; err = 0.044001746654480645, K = 1.378, L = 0.172,   = 0.85
68 iterations; err = 0.018932396860783074, K = 1.3752, L = 0.172,   = 0.85
69 iterations; err = 0.016092537331665602, K = 1.3727, L = 0.172,   = 0.85
70 iterations; err = 0.042655224785175916, K = 1.3791, L = 0.172,   = 0.85
71 iterations; err = 0.020076940449692104, K = 1.3761, L = 0.172,   = 0.85
72 iterations; err = 0.017065399382238278, K = 1.3736, L = 0.172,   = 0.85
73 iterations; err = 0.014505589474902658, K = 1.3714, L = 0.172,   = 0.85
74 iterations; err = 0.04400413046342444, K = 1.378, L = 0.172,   = 0.85
75 iterations; err = 0.018930370623180925, K = 1.3752, L = 0.172,   = 0.85
76 iterations; err = 0.016090815029703887, K = 1.3727, L = 0.172,   = 0.85
77 iterations; err = 0.04265668874184336, K = 1.3791, L = 0.172,   = 0.85
78 iterations; err = 0.02007569608652493, K = 1.3761, L = 0.172,   = 0.85
79 iterations; err = 0.017064341673546313, K = 1.3736, L = 0.172,   = 0.85
80 iterations; err = 0.014504690422514344, K = 1.3714, L = 0.172,   = 0.85
81 iterations; err = 0.04400489465795454, K = 1.378, L = 0.172,   = 0.85
82 iterations; err = 0.018929721057830307, K = 1.3752, L = 0.172,   = 0.85
83 iterations; err = 0.016090262899155716, K = 1.3727, L = 0.172,   = 0.85
84 iterations; err = 0.04265715805280945, K = 1.3791, L = 0.172,   = 0.85
85 iterations; err = 0.020075297172203577, K = 1.3761, L = 0.172,   = 0.85
86 iterations; err = 0.017064002596373085, K = 1.3736, L = 0.172,   = 0.85
87 iterations; err = 0.014504402206917177, K = 1.3714, L = 0.172,   = 0.85
88 iterations; err = 0.04400513964121222, K = 1.378, L = 0.172,   = 0.85
89 iterations; err = 0.018929512822061234, K = 1.3752, L = 0.172,   = 0.85
90 iterations; err = 0.01609008589875205, K = 1.3727, L = 0.172,   = 0.85
91 iterations; err = 0.042657308503152525, K = 1.3791, L = 0.172,   = 0.85
92 iterations; err = 0.02007516928941211, K = 1.3761, L = 0.172,   = 0.85
93 iterations; err = 0.01706389389600038, K = 1.3736, L = 0.172,   = 0.85
94 iterations; err = 0.014504309811600447, K = 1.3714, L = 0.172,   = 0.85
95 iterations; err = 0.04400521817723124, K = 1.378, L = 0.172,   = 0.85
96 iterations; err = 0.018929446066445088, K = 1.3752, L = 0.172,   = 0.85
97 iterations; err = 0.01609002915647828, K = 1.3727, L = 0.172,   = 0.85
98 iterations; err = 0.042657356734085106, K = 1.3791, L = 0.172,   = 0.85
99 iterations; err = 0.020075128293119304, K = 1.3761, L = 0.172,   = 0.85
100 iterations; err = 0.01706385904915142, K = 1.3736, L = 0.172,   = 0.85
101 iterations; err = 0.01450428019177874, K = 1.3714, L = 0.172,   = 0.85
```

```
102 iterations; err = 0.04400524335407985, K = 1.378, L = 0.172,   = 0.85
103 iterations; err = 0.018929424666123795, K = 1.3752, L = 0.172,   = 0.85
104 iterations; err = 0.016090010966205215, K = 1.3727, L = 0.172,   = 0.85
105 iterations; err = 0.04265737219581722, K = 1.3791, L = 0.172,   = 0.85
106 iterations; err = 0.020075115150647083, K = 1.3761, L = 0.172,   = 0.85
107 iterations; err = 0.017063847878050042, K = 1.3736, L = 0.172,   = 0.85
108 iterations; err = 0.014504270696342525, K = 1.3714, L = 0.172,   = 0.85
109 iterations; err = 0.04400525142520051, K = 1.378, L = 0.172,   = 0.85
110 iterations; err = 0.018929417805671367, K = 1.3752, L = 0.172,   = 0.85
111 iterations; err = 0.01609000513482073, K = 1.3727, L = 0.172,   = 0.85
112 iterations; err = 0.04265737715249407, K = 1.3791, L = 0.172,   = 0.85
113 iterations; err = 0.02007511093747172, K = 1.3761, L = 0.172,   = 0.85
114 iterations; err = 0.017063844296850927, K = 1.3736, L = 0.172,   = 0.85
115 iterations; err = 0.014504267652323266, K = 1.3714, L = 0.172,   = 0.85
116 iterations; err = 0.04400525401261701, K = 1.378, L = 0.172,   = 0.85
117 iterations; err = 0.018929415606367295, K = 1.3752, L = 0.172,   = 0.85
118 iterations; err = 0.016090003265412323, K = 1.3727, L = 0.172,   = 0.85
119 iterations; err = 0.042657378741491225, K = 1.3791, L = 0.172,   = 0.85
120 iterations; err = 0.020075109586824125, K = 1.3761, L = 0.172,   = 0.85
121 iterations; err = 0.017063843148800606, K = 1.3736, L = 0.172,   = 0.85
122 iterations; err = 0.014504266676480526, K = 1.3714, L = 0.172,   = 0.85
123 iterations; err = 0.044005254842083286, K = 1.378, L = 0.172,   = 0.85
124 iterations; err = 0.01892941490132083, K = 1.3752, L = 0.172,   = 0.85
125 iterations; err = 0.016090002666122816, K = 1.3727, L = 0.172,   = 0.85
126 iterations; err = 0.042657379250887306, K = 1.3791, L = 0.172,   = 0.85
127 iterations; err = 0.02007510915383759, K = 1.3761, L = 0.172,   = 0.85
128 iterations; err = 0.017063842780761895, K = 1.3736, L = 0.172,   = 0.85
129 iterations; err = 0.014504266363647655, K = 1.3714, L = 0.172,   = 0.85
130 iterations; err = 0.04400525510799125, K = 1.378, L = 0.172,   = 0.85
131 iterations; err = 0.018929414675299183, K = 1.3752, L = 0.172,   = 0.85
132 iterations; err = 0.016090002474004272, K = 1.3727, L = 0.172,   = 0.85
133 iterations; err = 0.04265737941418801, K = 1.3791, L = 0.172,   = 0.85
134 iterations; err = 0.020075109015031956, K = 1.3761, L = 0.172,   = 0.85
135 iterations; err = 0.017063842662777162, K = 1.3736, L = 0.172,   = 0.85
136 iterations; err = 0.014504266263360543, K = 1.3714, L = 0.172,   = 0.85
137 iterations; err = 0.04400525519323528, K = 1.378, L = 0.172,   = 0.85
138 iterations; err = 0.01892941460284181, K = 1.3752, L = 0.172,   = 0.85
139 iterations; err = 0.01609000241241554, K = 1.3727, L = 0.172,   = 0.85
140 iterations; err = 0.04265737946653858, K = 1.3791, L = 0.172,   = 0.85
141 iterations; err = 0.020075108970533995, K = 1.3761, L = 0.172,   = 0.85
142 iterations; err = 0.017063842624953862, K = 1.3736, L = 0.172,   = 0.85
143 iterations; err = 0.014504266231210705, K = 1.3714, L = 0.172,   = 0.85
144 iterations; err = 0.044005255220562534, K = 1.378, L = 0.172,   = 0.85
145 iterations; err = 0.0189294145796135, K = 1.3752, L = 0.172,   = 0.85
146 iterations; err = 0.016090002392671554, K = 1.3727, L = 0.172,   = 0.85
147 iterations; err = 0.042657379483320934, K = 1.3791, L = 0.172,   = 0.85
148 iterations; err = 0.02007510895626896, K = 1.3761, L = 0.172,   = 0.85
149 iterations; err = 0.017063842612828672, K = 1.3736, L = 0.172,   = 0.85
```

```
150 iterations; err = 0.014504266220904283, K = 1.3714, L = 0.172,   = 0.85
151 iterations; err = 0.04400525522932308, K = 1.378, L = 0.172,   = 0.85
152 iterations; err = 0.018929414572167014, K = 1.3752, L = 0.172,   = 0.85
153 iterations; err = 0.01609000238634195, K = 1.3727, L = 0.172,   = 0.85
154 iterations; err = 0.042657379488701075, K = 1.3791, L = 0.172,   = 0.85
155 iterations; err = 0.02007510895169573, K = 1.3761, L = 0.172,   = 0.85
156 iterations; err = 0.017063842608941338, K = 1.3736, L = 0.172,   = 0.85
157 iterations; err = 0.014504266217600259, K = 1.3714, L = 0.172,   = 0.85
158 iterations; err = 0.0440052552321315, K = 1.378, L = 0.172,   = 0.85
159 iterations; err = 0.018929414569780034, K = 1.3752, L = 0.172,   = 0.85
160 iterations; err = 0.01609000238431313, K = 1.3727, L = 0.172,   = 0.85
161 iterations; err = 0.04265737949042547, K = 1.3791, L = 0.172,   = 0.85
162 iterations; err = 0.020075108950230014, K = 1.3761, L = 0.172,   = 0.85
163 iterations; err = 0.017063842607695445, K = 1.3736, L = 0.172,   = 0.85
164 iterations; err = 0.014504266216541106, K = 1.3714, L = 0.172,   = 0.85
165 iterations; err = 0.04400525523303167, K = 1.378, L = 0.172,   = 0.85
166 iterations; err = 0.01892941456901487, K = 1.3752, L = 0.172,   = 0.85
167 iterations; err = 0.01609000238366276, K = 1.3727, L = 0.172,   = 0.85
168 iterations; err = 0.042657379490978364, K = 1.3791, L = 0.172,   = 0.85
169 iterations; err = 0.020075108949760168, K = 1.3761, L = 0.172,   = 0.85
170 iterations; err = 0.01706384260729621, K = 1.3736, L = 0.172,   = 0.85
171 iterations; err = 0.014504266216201822, K = 1.3714, L = 0.172,   = 0.85
172 iterations; err = 0.044005255233320106, K = 1.378, L = 0.172,   = 0.85
173 iterations; err = 0.01892941456876951, K = 1.3752, L = 0.172,   = 0.85
174 iterations; err = 0.01609000238345404, K = 1.3727, L = 0.172,   = 0.85
175 iterations; err = 0.04265737949115578, K = 1.3791, L = 0.172,   = 0.85
176 iterations; err = 0.0200751089496094, K = 1.3761, L = 0.172,   = 0.85
177 iterations; err = 0.01706384260716809, K = 1.3736, L = 0.172,   = 0.85
178 iterations; err = 0.014504266216092798, K = 1.3714, L = 0.172,   = 0.85
179 iterations; err = 0.04400525523341292, K = 1.378, L = 0.172,   = 0.85
180 iterations; err = 0.018929414568690683, K = 1.3752, L = 0.172,   = 0.85
181 iterations; err = 0.016090002383387203, K = 1.3727, L = 0.172,   = 0.85
182 iterations; err = 0.04265737949121262, K = 1.3791, L = 0.172,   = 0.85
183 iterations; err = 0.020075108949560994, K = 1.3761, L = 0.172,   = 0.85
184 iterations; err = 0.01706384260712679, K = 1.3736, L = 0.172,   = 0.85
185 iterations; err = 0.014504266216057715, K = 1.3714, L = 0.172,   = 0.85
186 iterations; err = 0.044005255233442675, K = 1.378, L = 0.172,   = 0.85
187 iterations; err = 0.01892941456866537, K = 1.3752, L = 0.172,   = 0.85
188 iterations; err = 0.016090002383365665, K = 1.3727, L = 0.172,   = 0.85
189 iterations; err = 0.04265737949123083, K = 1.3791, L = 0.172,   = 0.85
190 iterations; err = 0.02007510894954545, K = 1.3761, L = 0.172,   = 0.85
191 iterations; err = 0.01706384260711369, K = 1.3736, L = 0.172,   = 0.85
192 iterations; err = 0.014504266216046613, K = 1.3714, L = 0.172,   = 0.85
193 iterations; err = 0.044005255233452, K = 1.378, L = 0.172,   = 0.85
194 iterations; err = 0.0189294145686576, K = 1.3752, L = 0.172,   = 0.85
195 iterations; err = 0.016090002383359003, K = 1.3727, L = 0.172,   = 0.85
196 iterations; err = 0.0426573794912366, K = 1.3791, L = 0.172,   = 0.85
197 iterations; err = 0.020075108949540565, K = 1.3761, L = 0.172,   = 0.85
```

```
198 iterations; err = 0.01706384260710947, K = 1.3736, L = 0.172,   = 0.85
199 iterations; err = 0.01450426621604306, K = 1.3714, L = 0.172,   = 0.85
200 iterations; err = 0.04400525523345511, K = 1.378, L = 0.172,   = 0.85
201 iterations; err = 0.018929414568654934, K = 1.3752, L = 0.172,   = 0.85
202 iterations; err = 0.016090002383356783, K = 1.3727, L = 0.172,   = 0.85
203 iterations; err = 0.04265737949123838, K = 1.3791, L = 0.172,   = 0.85
204 iterations; err = 0.02007510894953901, K = 1.3761, L = 0.172,   = 0.85
205 iterations; err = 0.017063842607108137, K = 1.3736, L = 0.172,   = 0.85
206 iterations; err = 0.01450426621604195, K = 1.3714, L = 0.172,   = 0.85
207 iterations; err = 0.044005255233456, K = 1.378, L = 0.172,   = 0.85
208 iterations; err = 0.018929414568654046, K = 1.3752, L = 0.172,   = 0.85
209 iterations; err = 0.016090002383355895, K = 1.3727, L = 0.172,   = 0.85
210 iterations; err = 0.04265737949123927, K = 1.3791, L = 0.172,   = 0.85
211 iterations; err = 0.020075108949538345, K = 1.3761, L = 0.172,   = 0.85
212 iterations; err = 0.017063842607107693, K = 1.3736, L = 0.172,   = 0.85
213 iterations; err = 0.014504266216041506, K = 1.3714, L = 0.172,   = 0.85
214 iterations; err = 0.04400525523345644, K = 1.378, L = 0.172,   = 0.85
215 iterations; err = 0.018929414568653824, K = 1.3752, L = 0.172,   = 0.85
```

[ ]:
```
@time prim_exLab, res_exLab = MarketClearing(use_Fortran=false, tol = 1e-3,␣
↪exog_l = true);
```

[ ]:
```
@time prim_exLab_noSS, res_exLab_noSS  = MarketClearing(use_Fortran=false, tol␣
↪= 1e-3, ss = false, exog_l = true);
```

And the table:

[ ]:
```
L"\begin{tabular}{|l|c|c|c|c|c|c|}\hline",
    L"&\multicolumn{2}{c}{Benchmark Model} &\multicolumn{2}{c}{No risk,␣
↪$z^L$=$z^H=0.5$}",
    L"&\multicolumn{2}{c}{Exogenous labor, $\gamma=1$}\\\hline",
    L"capital, $K$ & ", out_res.K, " & ", res_noSS.K, " & ", res_noRisk.K, " &␣
↪", res_noRisk_noSS.K, " & ",
    res_exLab.K, " & ", res_exLab_noSS.K, " \\\hline",
    L"labor, $L$ & ", out_res.L, " & ", res_noSS.L, " & ", res_noRisk.L, " & ",␣
↪res_noRisk_noSS.L, " & ",
    res_exLab.L, " & ", res_exLab_noSS.L, " \\\hline",
    L"wage, $w$ & ", out_res.w, " & ", res_noSS.w, " & ", res_noRisk.w, " & ",␣
↪res_noRisk_noSS.w, " & ",
    res_exLab.w, " & ", res_exLab_noSS.w, " \\\hline",
    L"interest, $r$ & ", out_res.r, " & ", res_noSS.r, " & ", res_noRisk.r, " &␣
↪", res_noRisk_noSS.r, " & ",
    res_exLab.r, " & ", res_exLab_noSS.r, " \\\hline",
    L"pension benefit, $b$ & ", out_res.b, " & ", res_noSS.b, " & ", res_noRisk.
↪b, " & ", res_noRisk_noSS.b, " & ",
    res_exLab.b, " & ", res_exLab_noSS.b, " \\\hline",
```

```
   L"total welfare, $W$ & ", sum(out_res.val_fun.*out_res.F), " & ", res_noSS.
↪b, " & ", res_noRisk.b, " & ", res_noRisk_noSS.b, " & ",
   res_exLab.b, " & ", res_exLab_noSS.b, " \\\hline",
   L"cv(wealth) & ", Lambda(put_prim, out_res, W), " & ", Lambda(prim_noSS,␣
↪res_noSS, W), " & ", Lambda(prim_noRisk, res_noRisk, W),
   " & ", Lambda(prim_noRisk_noSS, res_noRisk_noSS, W), " & ",␣
↪Lambda(prim_exLab, res_exLab, W), " & ",
   Lambda(prim_exLab_noSS, res_exLab_noSS, W), " \\\hline \end{tabular}"
```