Quantitative Macroeconomics 2 PS2

Zhihang Liang, Tuba Şeker

Spring 2024

Exercise 2. Starting from Version 1.6 of your Problem Set. We want to compute some measures of wealth and income inequality.

1. **If you were using interp2, please update your code to get rid of it. Discretize your income process with more points. For instance, $N = 7$, or $N = 9$ if your code is fast enough. Optional (but strongly recommended): you may want to get rid of the loop on (s, k) and vectorize your code. Optional (2): you may want to switch to Rouwenhorst instead of Tauchen.**

The result of the computation is presented in figure 1.

We vectorize the golden search method using the following code:

```
function [x1,f1,indU] = golden(W_matrix, j,wage,r0,a,b)

global beta mu delta A alpha s N prob  kk kap v dist nkap

    W_func = griddedInterpolant(kap,W_matrix(j,:));


% lower the tolerance compared to previous case to increase the speed
tol = 0.00001;

alpha1 = (3-sqrt(5))/2;
alpha2 = (sqrt(5)-1)/2;
d  = b-a;
x1 = a+alpha1*d;
x2 = a+alpha2*d;


c1 = s(j)*wage + (1+r0).*kap -x1;
util1 = (c1.^(1-mu)-1)./(1-mu);
util1(c1<0)= -inf;
f1 = util1 + W_func(x1);


c2 = s(j)*wage + (1+r0).*kap -x2;
util2 = (c2.^(1-mu)-1)./(1-mu);
util2(c2<0)= -inf;
f2 = util2 + W_func(x2);

d = alpha1*alpha2.*d;
while max(d)>tol
    d = d.*alpha2;
    x1_temp = x1;
    x2_temp = x2;
    x1 = (x1_temp-d).*(f1-f2>0) + x2_temp.* (f1-f2<=0);
    x2 =  x1_temp.*(f1-f2>0) + (x2_temp+d).*(f1-f2<=0);

    c1 = s(j)*wage + (1+r0).*kap -x1;
    util1 = (c1.^(1-mu)-1)./(1-mu);
    util1(c1<0)= -inf;
    f1 = util1 + W_func(x1);


    c2 = s(j)*wage + (1+r0).*kap -x2;
    util2 = (c2.^(1-mu)-1)./(1-mu);
    util2(c2<0)= -inf;
    f2 = util2 + W_func(x2);

end

% Return the larger of the two
 x1(f2>f1) = x2(f2>f1);
 f1(f2>f1) = f2(f2>f1);
 indU = zeros(1,nkap);
     for k = 1:nkap
         indU(k) = find(kap>=x1(k),1,'first');
     end
end
```

There are several features of our golden search function we would like to highlight.

Firstly, we fix j, the index for technology, each time we compute the interpolation. We argue that to exploit both the benefit of vectorziation and one-dimensional interpolation, this is sufficient. To compute the value of the updated value function $u+W$ using one-dimensional interpolation, we need to fix the technology level. However, if we vectorize everything, we need to compute a long vector of $u+W$, for each element of which we need to learn about their technology level in order to compute one-dimensional interpolation. Thus, we need to add an additional condition for each element, adding to the complexity of the code.

Secondly, we stop using anonymous function an input for the golden search function. Instead, we use the matlab built-in `griddedInterpolant` function inside the golden function to compute interpolation. Using anonymous function slows down matlab by a lot. Using `griddedInterpolant` is more efficient than the `interpl1` or self-made interpolation function, in the sense that we don't need to recalculate interpolation in each iteration. We also tried manually interpolating when using the vectorized golden search function:

```
for k=1:nkap
    ind2(k) = find(kap>=x2(k),1,'first');
end
f2 = util2 + (x2-kap(ind2-1)) ./ (kap(ind2)-kap(ind2-1)).* W_matrix(j,ind2)+
(kap(ind2)-x2) ./ (kap(ind2)-kap(ind2-1)).* W_matrix(j,ind2-1);
```

which enormously slows down the code, as we need to create a loop for the asset each time we evaluate the continuation value. In fact, our code takes only 4 seconds to run, and we manually interpolate things, it takes over 100 seconds to run.

Thirdly, we adjust the computation of the distribution. We assign probability for the asset policy to go to the upper grid and the lower grid, since the asset policy function is now off-grid with interpolation. Note that, in figure 1, the distribution becomes smoother.

Although we don't implement a complete vectorization, our code is sufficiently fast due to the fact that N is small. With $N=5$ and $nkap=415$, the code takes 4 seconds to run. We keep $N=5$ because when we change $N=7$, the code becomes unstable since we update the interest rate guess $r0$ by using $r0=0.8r0+0.2r1$. On the other hand, when we use bisection, the code becomes more stable.

The result shows that $k^*=6.3006$ and $r^*=0.0235$

2. **Sort households by wealth, and compute the share of total wealth held by each wealth quintile. Compute mean income by wealth quintile. What can you say about the correlation between wealth and income?**

Here, we sort wealth and create five quantiles. Then, calculate mean income by using the same wealth quantiles. The result is shown in figure 2. We get a very high correlation between income and wealth: 94%. This result is expected since the ones who work and earn higher incomes are the same group of people who have higher savings motives.

3. **Now decrease the autocorrelation parameter of the income process, from 0.9 to 0.8. What can you say about wealth inequality? About the correlation between wealth and income?**

Once we decrease the persistence, the wealth inequality decreases. When $\rho = 0.8$, the wealth distribution is shown in figure 3. While the top quantile was capturing 60% of wealth previously when $\rho = 0.9$, it now captures slightly above 50% of the total wealth. Similarly, lower quantiles now accumulate more wealth since any bad productivity shock will last shorter and enable these people to save more. High-skilled workers face a higher probability to have lower wage in the future, decreasing their wealth level.

Confirming these observations, we observe that the correlation between wealth and income decreases from 94% to 89%.

Exercise 3. Starting from Exercise 2 of your Problem Set, go back to $N = 5$, but replace the Value Function Iteration block by an Endogenous Grid Method, starting with a guess on the policy function for c.

In the endogenous grid point method, we now construct grids for the next period asset, a', rather than a, and then look for convergence of the policy function c. Steps are:

1. Start from any guess of policy function $c = g_0(a, \epsilon)$
2. Calculate the RHS of the EE: $\beta(1+r)\sum_\epsilon (g_0(a,\epsilon))^{-\mu}$
3. Obtain consumption today as a function of a' and ϵ where both are on the grid: $c(a', \epsilon) = [RHS(a', \epsilon)]^{-\frac{1}{\sigma}}$
4. By using budget constraint, obtain $a(a', \epsilon)$: $a = \frac{c(a',\epsilon)+a'-\epsilon w}{1+r}$
5. Get c_1 by interpolation since now we obtain a grid $\{(c(a', \epsilon), a', a(a', \epsilon))\}$. Then, we update g_1 and check for convergence $g_1 \approx g_0$:

$$interp(a(a', \epsilon), c(a', \epsilon), a) \equiv g_1(a, \epsilon)$$

Our code is presented below:

```
while test>1e-8
    iter= iter+1;
    %Implied consumption:
    c_imp = (beta*(1+r0)*prob*(c0).^(-mu)).^(1/-mu);

    %Implied savings:
    % Given c_imp, a', and productivity, obtain a by budget constraint
    a_imp = (kap+c_imp-wage*s')/(1+r0);


    % Make sure a (savings) on the grid
    for j=1:N
        kk(j,:) = interp1(a_imp(j,:),kap,kap, 'linear','extrap');
    end

    % Borrowing constraint bounds to prevent error in the measure while
    % calculating the measure
    kk(kk<minkap) = minkap;
    kk(kk>maxkap) = maxkap;

    % Consumption policy function
    c_egm = wage*s'+(1+r0)*kap-kk;

    % Distance measure
    test = max(max(abs(c_egm-c0)));
    c0 = c_egm;

end
```

Endogenous grid point method is very handy in the sense that it can accommodate vectorization without effort. There are two points to note. Firstly, since now we interpolate the implied asset to get the asset policy function, we need to specify the extrapolation method in case of the asset grid is outside the range of `a_implied`. Secondly, we need to restrict the minimum of the asset policy function to be no less than the borrowing constraint.
EGM is very fast. It only takes 2.45 seconds to run.
The result shows that $k^* = 6.3058$ and $r^* = 0.0234$, which is close to the result in exercise 2.

Exercise 4. We add a government to the economy of Problem Set I. The government has to finance some spending G, non-valued by the consumer, with a flat tax τ on total income.

1. **Add the government in your code. Find the equilibrium interest rate $r^\star$ and tax rate $\tau^\star$ when $G = 0.1$. Hint: you need only one loop.**

Once we add the government, we first modify our loop as:

```
while (testr>0.001)
    wage = (1-alpha)*(A*(alpha/(r0+delta))^alpha)^(1/(1-alpha));
    k0=((r0+delta)/(alpha*A*labor^(1-alpha)))^(1/(alpha-1));
    tau0 = G/(r0*k0+wage*labor);
    k1= aiyagari4_1(r0,tau0,wage);
    r1= alpha*A*max(0.001,k1)^(alpha-1)*labor^(1-alpha)-delta;

    %Q4: Implied interest rate:
    r0 = 0.8*r0 + 0.2*r1;

    testr=abs(r1-r0);
    %updating the interest rate
    if k1>k0
        maxrate=r0;
    else
        minrate=r0;
    end

end
```

We use EGM and the code runs in 0.352009 seconds with 415 grid points. We get the following output:

k^*	r^*	τ^*
6.0499	0.0267	0.0655

We set $G = 0.1$. The result is not that different from the results in exercise 2 and exercise 3, except that capital is lower and interest rate is higher. The lump-sum tax decreases the saving level, thus increasing the interest rate.

2. **Now, calibrate the government spending to target a government-to-output ratio of 15%. Hint: you need only one loop.**

   Note: you can use VFI or EGM.

Here, we add production and calibrated G/Y ratio inside the loop.

```
while (testr>0.001 )
    wage = (1-alpha)*(A*(alpha/(r0+delta))^alpha)^(1/(1-alpha));
    k0=((r0+delta)/(alpha*A*labor^(1-alpha)))^(1/(alpha-1));
    Y = A* k0^(alpha)* labor^(1-alpha);
    G = targetg*Y;
    tau0 = G/(r0*k0+wage*labor);
    k1= aiyagari4_1(r0,tau0,wage);
    r1= alpha*A*max(0.001,k1)^(alpha-1)*labor^(1-alpha)-delta;

    %Q4: Implied interest rate:
    r0 = 0.8*r0 + 0.2*r1;
    G = 0.8*G + 0.2*(targetg*Y);


    testr=abs(r1-r0);
end
```

By EGM, it takes 1.417657 seconds code to run, and we obtain:

k^*	r^*	τ^*
5.5537	0.0339	0.2053

Once we obtain the output, τ is adjusted to match the given G/Y ratio, thus we obtain a much higher ratio for tax rate, $\tau \approx 21\%$. This is expected since targeted government spending is higher than in the previous case.

From Figure 6, we observe that once we calibrate $G/Y = 0.15$, we closely follow the patterns of the case of no government, as in question 3.

Exercise 5. We want to extend the economy (without a government) to incorporate endogenous labor supply. Let $\varphi = 1/0.4$. The preferences of the household are:

$$u(c,n) = \frac{c^{1-\mu}}{1-\mu} - B\frac{n^{1+\varphi}}{1+\varphi}$$

1. **We first solve for endogenous labor supply when we solve for a' using a grid search (that is, version 1.4 of Problem Set I). For each guess of prices, use bissection to compute, for each (a, ε, a') on your grid for assets and productivity, a function $\hat{n}(a, \varepsilon, a')$ which finds the optimal labor supply given assets a, productivity ε and new asset position a'. Then, use this function $\hat{n}$ and find the optimal asset policy function; compute the measure; and solve for your equilibrium. Find a value for B such that average hours worked are approximately equal to 0.3. Plot policy function for labor as a function of a and ε.**

We vectorize the process to find $\hat{n}$:

```
for j=1:N
    % Initialise variables
        % Bounds
        nlow = (kap-(1+r0)*kap')/(wage*s(j)) + 1e-4;
        nup = ones(nkap,nkap);
        % Loop criterium
        maxtestn = 1;
    % Loop
    while (maxtestn>0.001)
        % Update guess
            n0 = 0.5*(nup-nlow)+nlow;
        % Evaluate the FOC
            f=focn(s(j),n0);
        % Test distance: how far are we from 0 (satisfied FOC)
            testn=abs(f);
        % Update bounds
            nup(f>0)=n0(f>0);
            nlow(f<0)=n0(f<0);
        % If nlow goes to one, we are looking at a' too high
            nlow(1-nlow<1e-3) = 1;
            nup(1-nlow<1e-3) = 1;
            testn(1-nlow<1e-3) = 0;
        % Update max distance
            maxtestn = max(max(testn));
    end
    nopt(:,:,j)=max(n0,0);
    % Rows: a. Columns: a'. 3rd dimension: current productivity.
end
```

The code runs in 85.957922 seconds and find that approximately $B = 150$, we obtain average hours worked are approximately equal to 0.3. Note that we also created a loop for B to keep track of which value of it satisfies the given average hours worked equal to 0.3, and we obtained $B = 153.1250$.

k^*	r^*	n^*
1.5483	0.0257	0.2993

Figure 7 shows that the policy function for consumption is no longer smooth, due to the fact that we don't use interpolation here. Labor supply is relatively higher for higher skill levels, and as asset holdings increase, labor supply decreases for all skill types.

Note that compared with the inelastic labor supply case, the capital level is lower due to the disutility of labor, while the interest rate is approximately the same. Now, labor demand can adjust according to the ratio of w/r, thus guaranteeing a more productive environment.

2. **Starting from your code with a Golden search (version 1.6 of Problem Set I), now interpolate in your Golden code not only the value function (or continuation value) but also the policy function for labor (use the $\hat{n}$ that you computed above to interpolate).**

We fix B to 150 now. The code runs faster , which is 40.802519 seconds.

k^*	r^*	n^*
1.5736	0.0248	0.3005

In Figure 8, we get now smoother policy function and labor supply. Similar to exercise 2, we interpolate inside the golden search function and:

```
vf_int=(x2-kap(ind2-1)) ./ (kap(ind2)-kap(ind2-1)) .* v(:,ind2)+
(kap(ind2)-x2) ./ (kap(ind2)-kap(ind2-1)) .* v(:,ind2-1);

n2_int = (x2-kap(ind2-1)) ./ (kap(ind2)-kap(ind2-1)).* nopt(sub2ind(size(nopt),1:nkap,ind2)) +
(kap(ind2)-x2) ./ (kap(ind2)-kap(ind2-1)).* nopt(sub2ind(size(nopt),1:nkap,ind2-1));
```

```
        cih = prodwage*n2_int + (1+r0)*kap;
        u2 = util(max(cih-x2,0.01),n2_int);
        v2 =  sum(prob' .* vf_int);
        f2 = u2 + beta*v2;
```

We still vectorize the process given the productivity levels. Inside the golden search, we need to create a loop to search the index of each element whenever we want to evaluate things, which might slow down the process:

```
    for k=1:nkap
        ind2(k) = find(kap>=x2(k),1,'first');
    end
```

The other approach is to use `griddedInterpolant` plus vectorization, as in exercise 2. In our function, we manually interpolate as shown above.

Figures

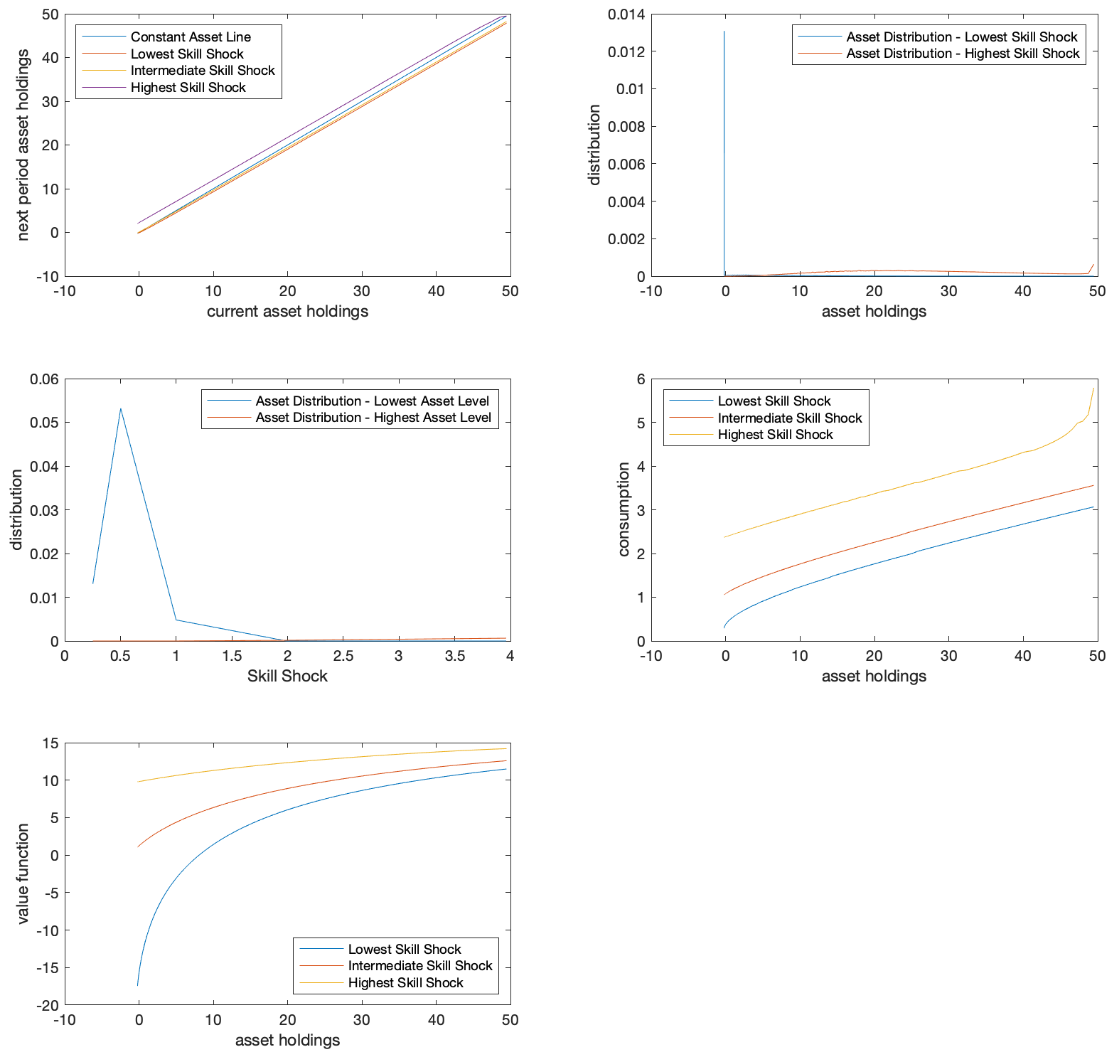

Figure 1: Exercise 2

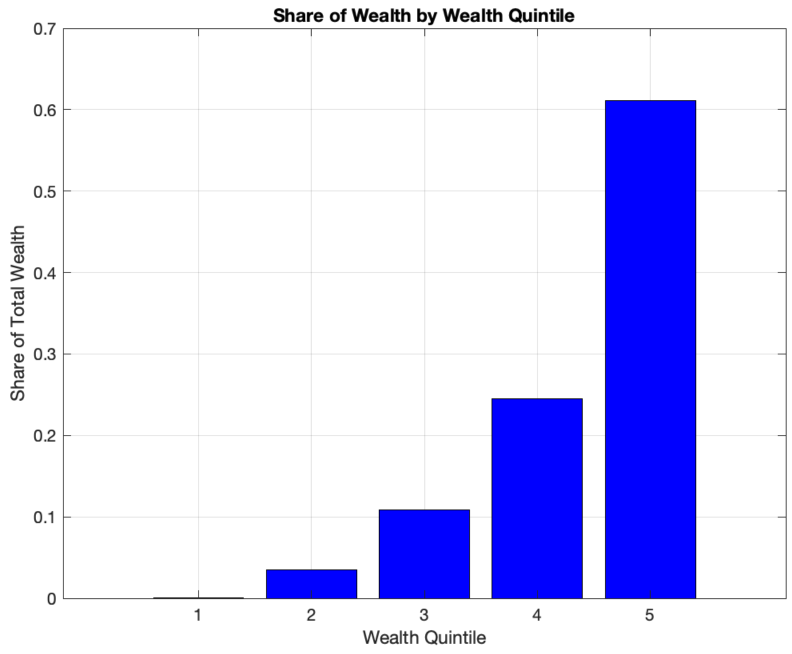

Figure 2: Wealth Inequality, $\rho = 0.9$

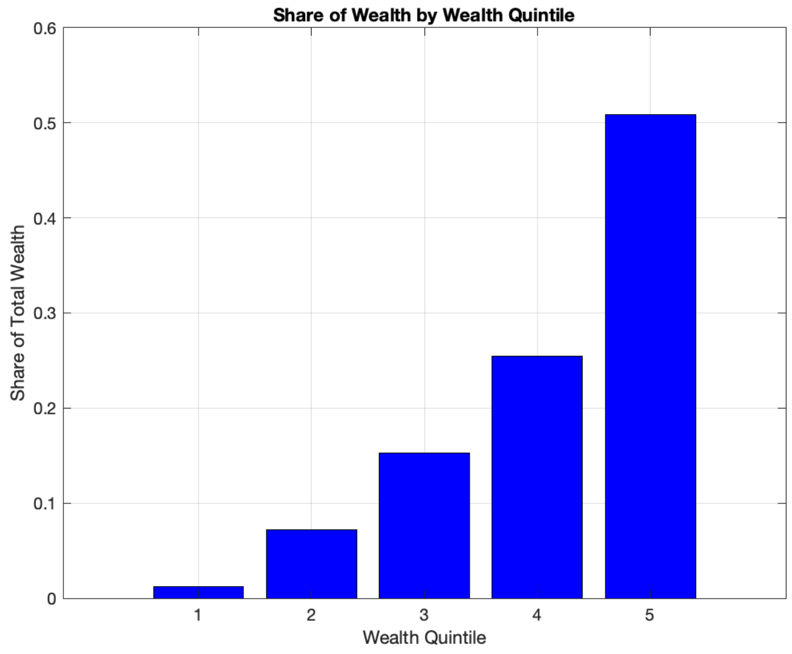

Figure 3: Wealth Inequality, $\rho = 0.8$

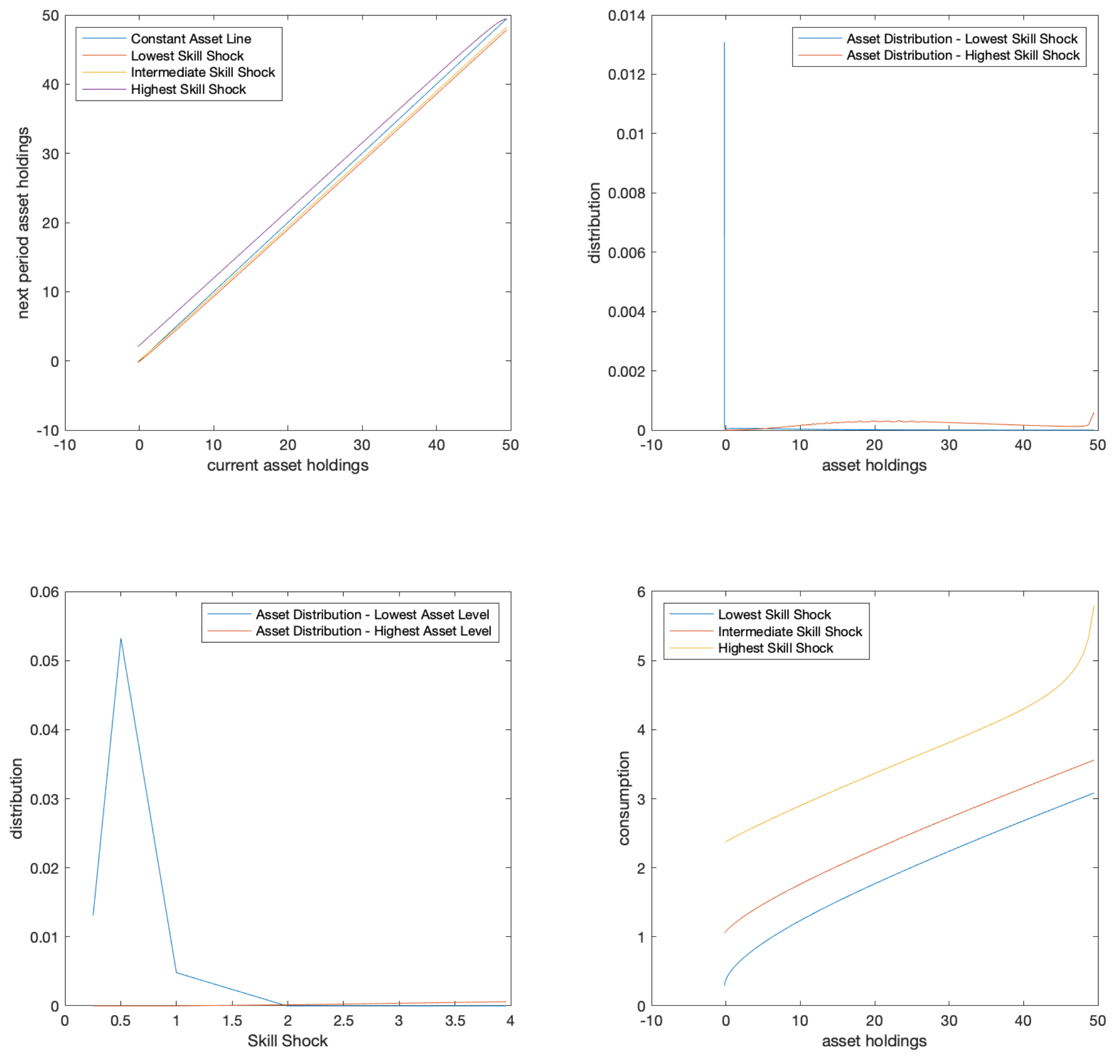
Constant Asset Line
Lowest Skill Shock
Intermediate Skill Shock
Highest Skill Shock
next period asset holdings
current asset holdings
Asset Distribution - Lowest Skill Shock
Asset Distribution - Highest Skill Shock
distribution
asset holdings
Asset Distribution - Lowest Asset Level
Asset Distribution - Highest Asset Level
distribution
Skill Shock
Lowest Skill Shock
Intermediate Skill Shock
Highest Skill Shock
consumption
asset holdings

Figure 4: EGM

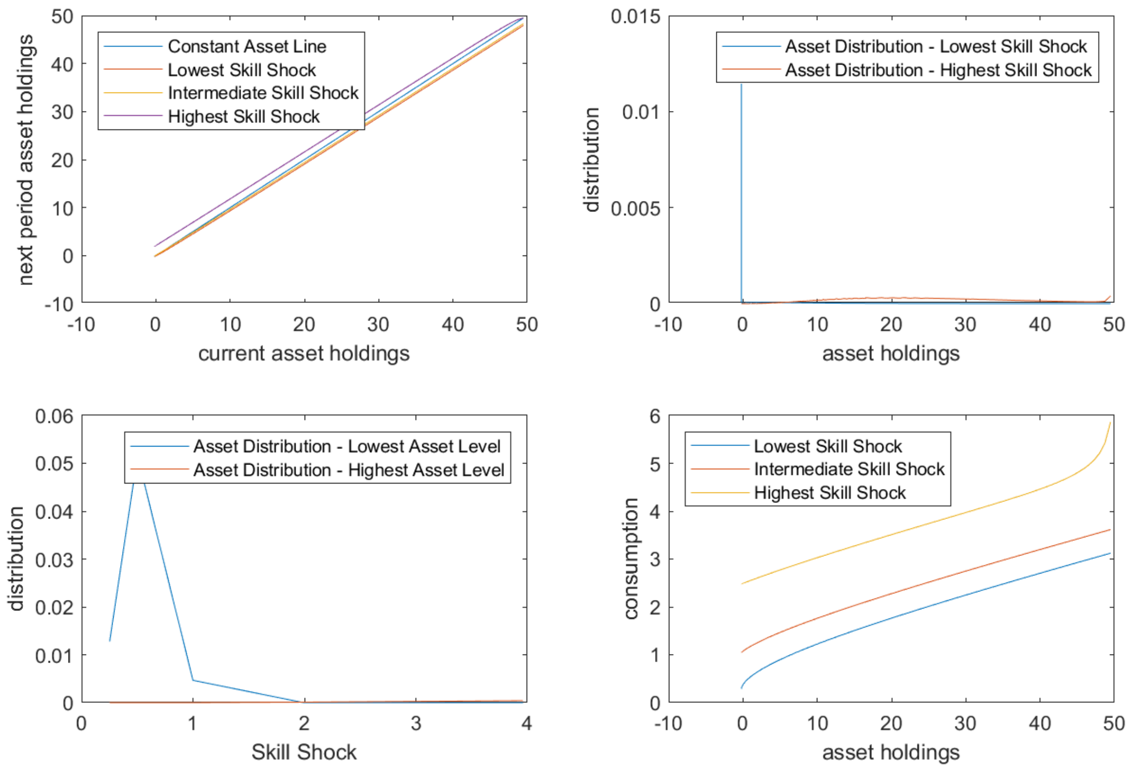

Figure 5: EGM with government

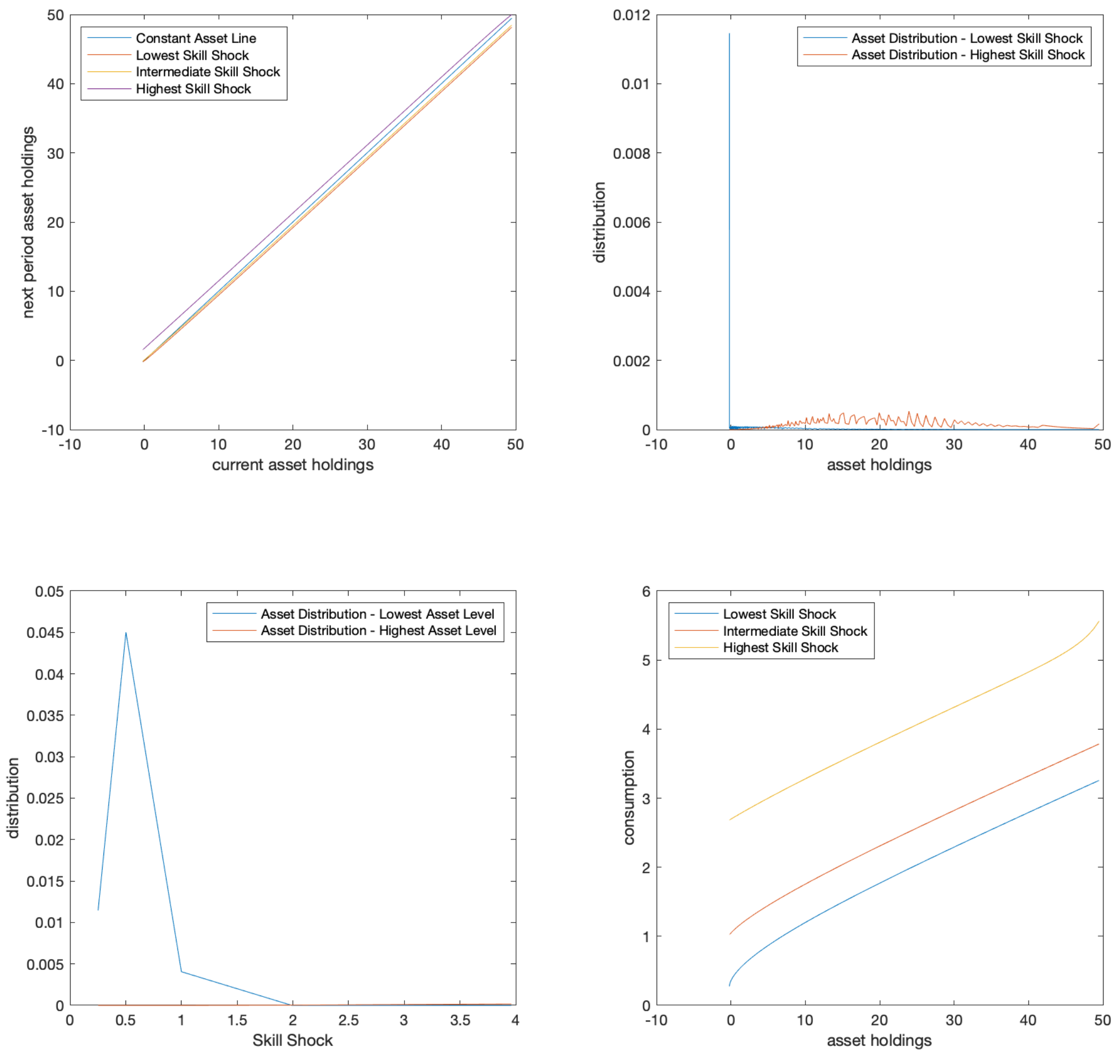

Figure 6: EGM with government, G/Y=0.15

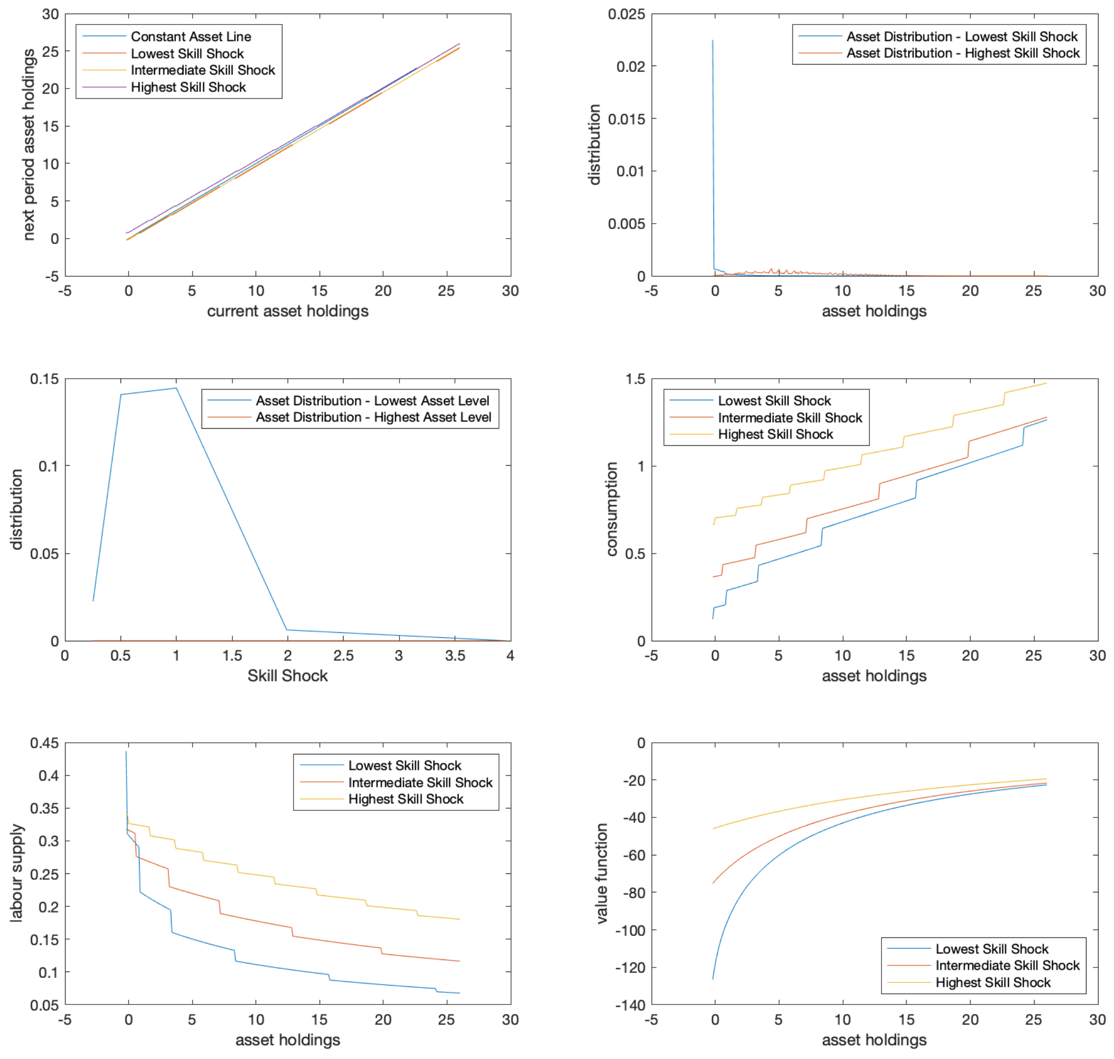

Figure 7: VFI, Endogenous Labor Supply

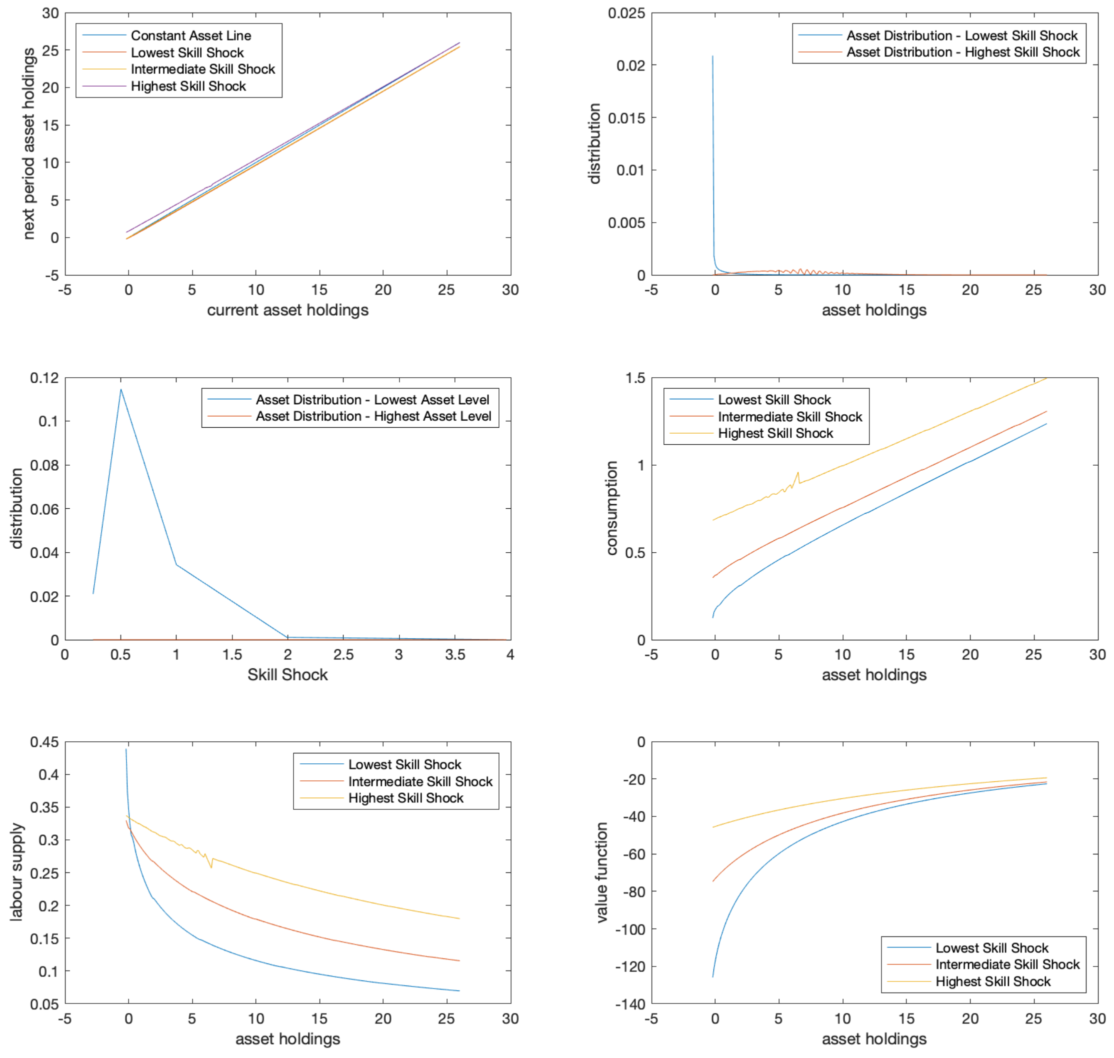

Figure 8: VFI, Endogenous Labor Supply, Golden updated