ECON 634 Problem Set 4

Mingyang Li

November 16, 2017

1 The first-order conditions for the firm are

$$\frac{\partial \pi(K_t, N_t; w_t, r_t)}{\partial K_t} = 0,$$

$$\frac{\partial \pi(K_t, N_t; w_t, r_t)}{\partial N_t} = 0.$$

That is,

$$F_{K_t}(K_t, N_t) - r + (1-\delta) = 0,$$
$$F_{N_t}(K_t, N_t) - w_t = 0.$$

Therefore,

$$r_t = \alpha \left(\frac{K}{N}\right)_t^{\alpha-1} + (1-\delta),$$

$$w_t = (1-\alpha)\left(\frac{K}{N}\right)_t^{\alpha}.$$

2 The households' recursive problem is

$$\max_{\{c_t\}_{t=0}^{\infty}} \mathbb{E}_0 \left[\sum_{t=0}^{\infty} \beta^t \frac{c_t^{1-\sigma}}{1-\sigma}\right]$$

$$s.t.\ c_t + a_{t+1} = z_t w_t \bar{l} + r_t a_t$$

$$\ln z_{t+1} = \rho \ln z_t + \varepsilon_t$$

$$l_t = \bar{l} = 1$$

$$a_{t+1} \in \Gamma(s_t, a_t)$$

$$a_{t+1} \geq \underline{a},\ \forall t,\ \text{and } a_0 \text{ given.}$$

6 The steady state interest rate $r = 1.0101$, which is equal to the complete markets interest rate $r^{CM} = 1/\beta = 1.0101$.
According to my results, there is no big difference in the Lorenz Curve between Aiyagari and Huggett.

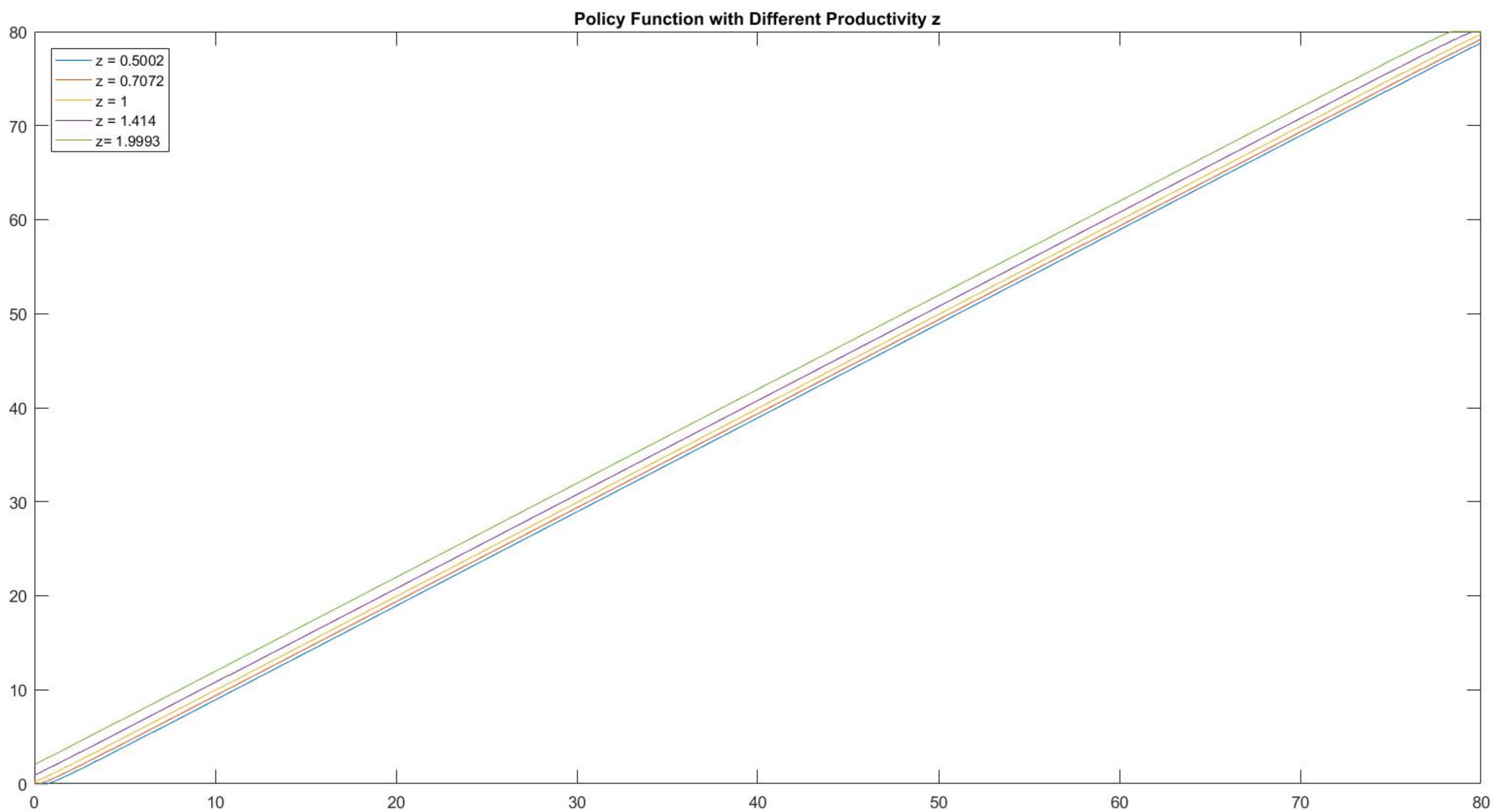

Figure 1: Policy Functions

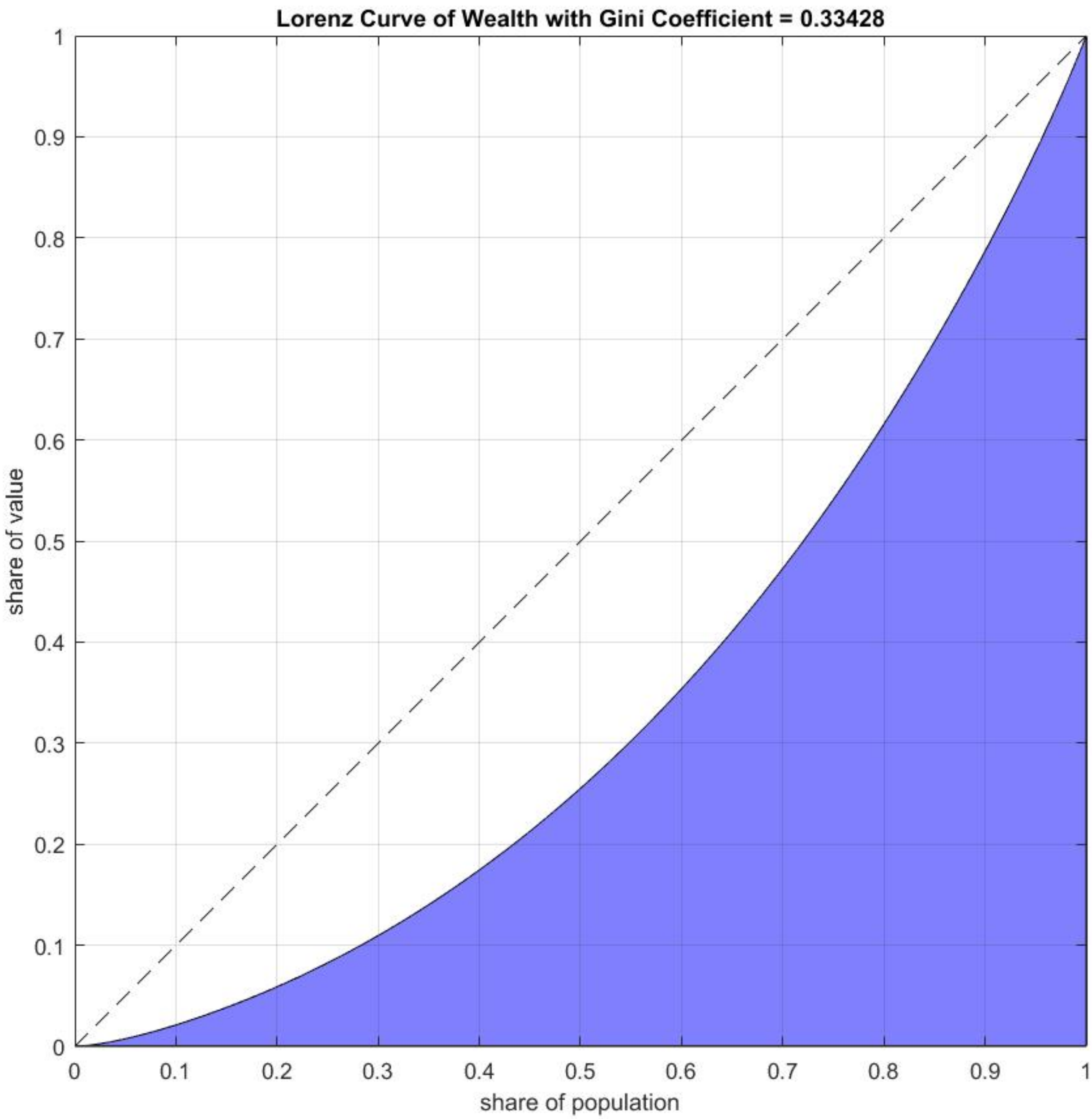

Figure 2: Lorenz Curve of Aiyagari Model

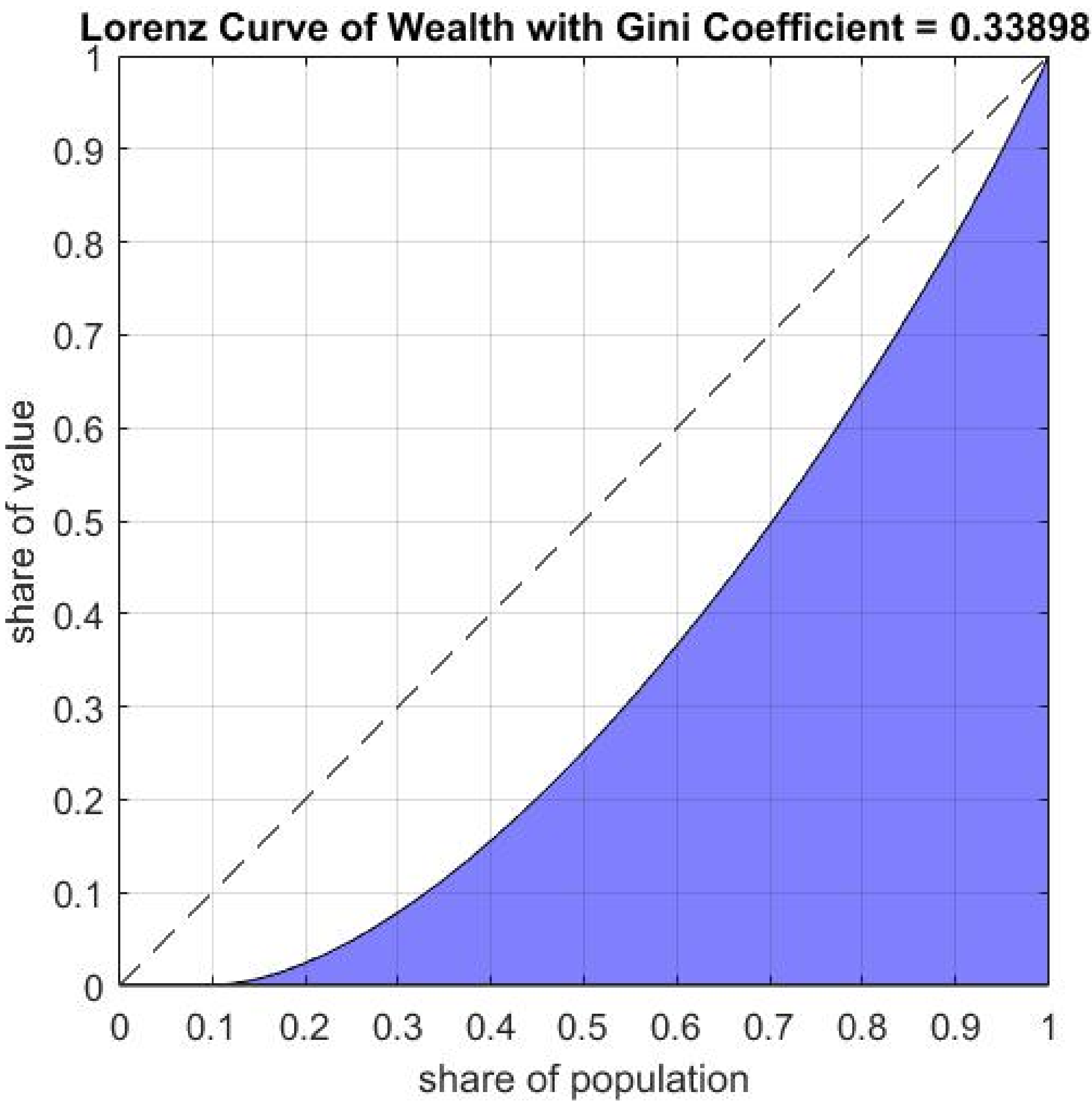

Figure 3: Lorenz Curve of Huggett Model

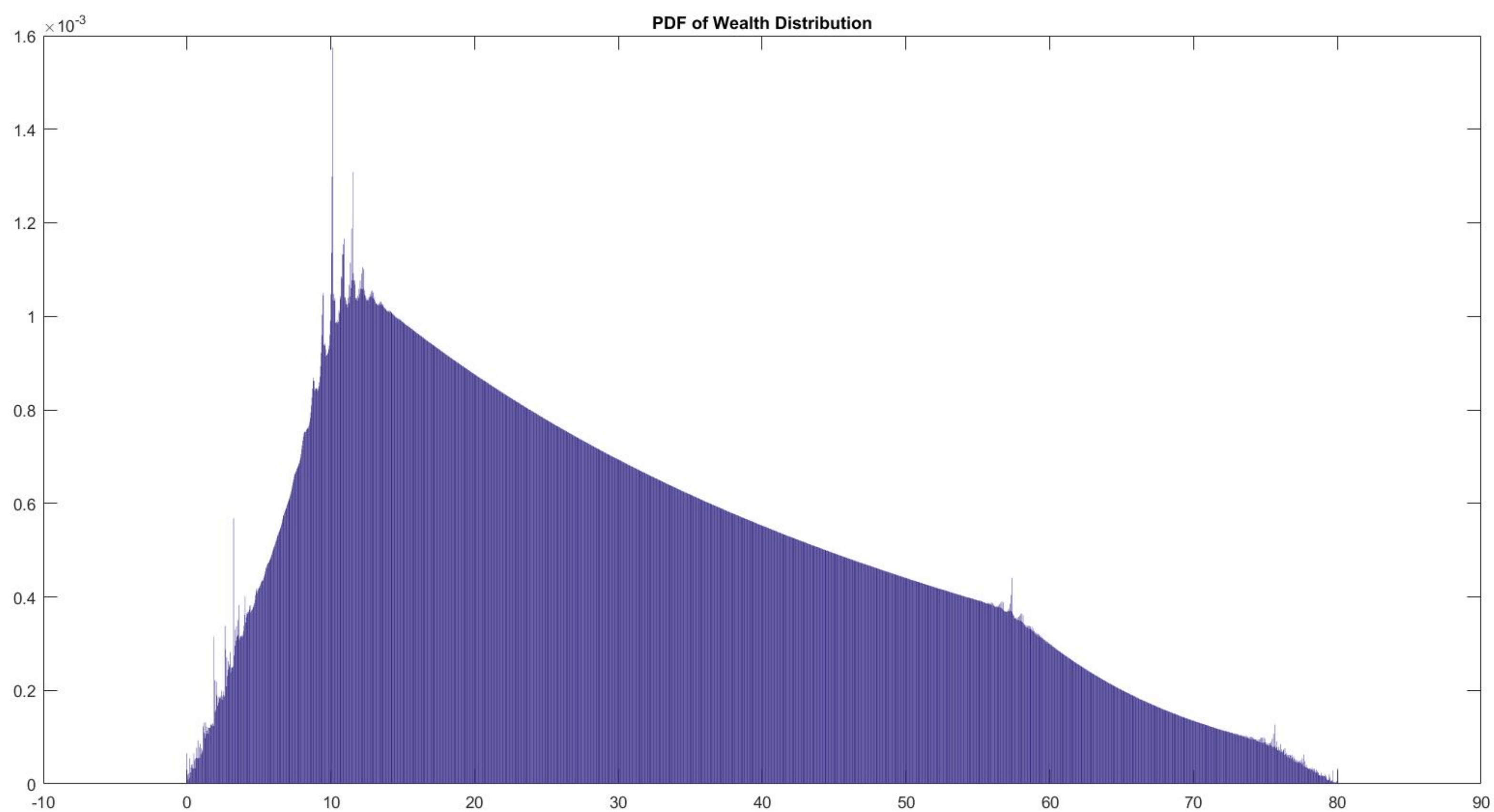

Figure 4: Wealth Distribution

7 I set up the following initial conditions for asset and capital:

```
a_lo = 0; a_hi = 80; num_a = 2000; K_max = 50; K_min = 20; abs(K_tol) >= .01.
```

Using Value Function Iteration, I got a Gini coefficient of 0.33374 and elapsed time is 7506.545284 seconds.
Using Policy Function Iteration, I got a Gini coefficient of 0.33428 and elapsed time is 8256.467155 seconds.
The PFI approach is slower than the VFI approach!
I didn't manage to solve the interpolation parts...

Appendices

A The main program PS4.m

```
% PROGRAM NAME: ps4aiyagari
clear, clc
tic

% PARAMETERS
alpha = 1/3; % Cobb-Douglas production function parameter
beta = .99; % discount factor
sigma = 2; % coefficient of risk aversion
rho = 0.5; % coefficient of log(z_t)
sigma_z = 0.2; % standard deviation of epsilon in AR-1 of log(z_t)
delta = 0.025; % depreciation rate

% Input argument for Tauchen's method
num_z = 5;
m = 3;
[lnz, PI] = TAUCHEN(num_z, rho, sigma_z, m);
z = exp(lnz');

% Find the invariant PI_in
% % Prof. Kuhn's code
% [eigvecs, eigvals] = eig(PI');
% PI_star = eigvecs(:,1) ./ sum(eigvecs(:,1));
% % invariant distribution is the first eigenvector of PI normalized to one

% Mingyang's method, same result for aggregate labor
tol = 1;
% i = 0; % iteration counter
while (tol > 1e-6)
    PI_in = PI * PI;
    tol = max(max(abs(PI - PI_in)));
    PI = PI_in;
    %     i = i + 1
end
% PI_in
N_s = z * PI_in(1, :)'; % compute the aggregate effective labor supply
N_d = N_s;

% ASSET VECTOR
% Why do we have the lower and upper bounds of asset?
```

```
a_lo = 0; % lower bound of grid points
a_hi = 80; % upper bound of grid points
% How to decide the size of asset vector?
num_a = 2000;
a = linspace(a_lo, a_hi, num_a); % asset (row) vector

% % what is aggregate labor? by Prof. Kuhn.
% agg_L = z * PI_star;

% Guess a value for K. Where is this guess from? Trial and error.
K_max = 50;
K_min = 20;

% ITERATE OVER ASSET PRICES
aggsav = 1 ;
K_tol = 1;
while abs(K_tol) >= .01

    K_guess = (K_min + K_max) / 2;
    % Calculate the factor prices
    r = alpha * (K_guess / N_d) ^ (alpha - 1) + (1 - delta);
    w = (1 - alpha) * (K_guess / N_d) ^ alpha;

    % CURRENT RETURN (UTILITY) FUNCTION
    cons = bsxfun(@minus, r * a', a);
    cons = bsxfun(@plus, cons, permute(z * w, [1 3 2])); % permute(z, [1 3 2]) * w?
    ret = (cons .^ (1-sigma)) ./ (1-sigma); % current period utility
    ret(cons<0) = -Inf;

    % INITIAL VALUE FUNCTION GUESS
    v_guess = zeros(num_z, num_a);

    vfi_method = 1;
    % VALUE FUNCTION ITERATION
    if vfi_method == 1
        grid_search
    elseif vfi_method == 2
        pol_fn_iter
    elseif vfi_method == 3
        vfi_interpolation
    else
        grid_search
    end

    % KEEP DECSISION RULE
    pol_fn = a(pol_indx);
```

```
    % SET UP INITITAL DISTRIBUTION
    Mu = ones(num_z, num_a); % alternative initial guess: same mass in all states
    Mu = Mu / sum(Mu(:)); % normalize total mass to 1

    % ITERATE OVER DISTRIBUTIONS
    % loop over all non-zeros states
    mu_tol = 1;
    while mu_tol > 1e-08
        [z_ind, a_ind] = find(Mu > 0); % find non-zero indices
        MuNew = zeros(size(Mu));
        for ii = 1:length(z_ind)
            apr_ind = pol_indx(z_ind(ii), a_ind(ii));
            MuNew(:, apr_ind) = MuNew(:, apr_ind) + ...
                (PI(z_ind(ii), :) * Mu(z_ind(ii), a_ind(ii)) )';
        end
        mu_tol = max(abs(MuNew(:) - Mu(:)));
        Mu = MuNew ;
    end

    % CHECK AGGREGATE DEMAND
    aggsav = sum( pol_fn(:) .* Mu(:) ); % Aggregate future assets
    K_tol = aggsav - K_guess;
    if K_tol > 0 ;
        K_min = K_guess ;
    end ;
    if K_tol < 0;
        K_max = K_guess ;
    end ;

    display (['K = ', num2str(K_guess)])
    display (['Aggregate desired wealth = ', num2str(aggsav)]);
    display (['New Kmin is ', num2str(K_min), ', new Kmax is ', num2str(K_max)]);
    display (['New K is ', num2str((K_max + K_min)/2)]);
    display (['Tolerance is ', num2str(K_tol)]);
    display (' ') ;

end

% interest rate in complete market
r_cm = 1/beta;

% plot the value function
plot(a, vfn)
legend('z = 0.5002','z = 0.7072','z = 1','z = 1.414','z= 1.9993','Location','northwest')
title('Value Function with Different Productivity z')
```

```
% plot the policy function
figure
plot(a, pol_fn)
legend('z = 0.5002','z = 0.7072','z = 1','z = 1.414','z= 1.9993','Location','northwest')
title('Policy Function with Different Productivity z')

% FIND TOTAL WEALTH DISTRIBUTION AND GINI
agg_wealth = sum(Mu,1) * a'; % wealth is asset holdings
wealth_dist = [[Mu(1,:), Mu(2,:), Mu(3,:), Mu(4,:), Mu(5,:)]; [a, a, a, a, a]]';
[~, ordr] = sort(wealth_dist(:,2), 1);
wealth_dist = wealth_dist(ordr,:);

% see formula on wikipedia for computation of gini in discrete distributions:
pct_dist = cumsum( (wealth_dist(:,2) ./ agg_wealth) .* wealth_dist(:,1) );
gini_coe = 1 - sum( ([0; pct_dist(1:end-1)] + pct_dist) .* wealth_dist(:,1) );
display (['Gini coefficient of ', num2str(gini_coe)]);

% plot the CDF of wealth distribution
figure
plot(wealth_dist, pct_dist)
title('CDF of Wealth Distribution')

% plot the PDF of wealth distribution
mu = sum(Mu);
figure
bar(a,mu)
title('PDF of Wealth Distribution')

% Plot the Lorenz curves and compute the Gini coefficients
Mu_trans = Mu';
pop = [Mu_trans(:, 1); Mu_trans(:, 2); Mu_trans(:, 3); Mu_trans(:, 4); Mu_trans(:, 5)];
val_wealth = repmat(a', [num_z, 1]); % [a'; a'; a'; a'; a'];
val_wealth(val_wealth < 0) = 0;
figure
[gini_wealth, l_wealth] = gini(pop, val_wealth, true);
title(['Lorenz Curve of Wealth with Gini Coefficient = ', num2str(gini_wealth)])

toc
```

B The function file grid_search.m

```
% value function iteration (grid search)
v_tol = 1;
while v_tol >.0001;
```

```
  % CONSTRUCT TOTAL RETURN FUNCTION
  v_mat = ret + beta * ...
      repmat(permute(PI * v_guess, [3 2 1]), [num_a 1 1]);

  % CHOOSE HIGHEST VALUE (ASSOCIATED WITH a' CHOICE)
  [vfn, pol_indx] = max(v_mat, [], 2);
  vfn = permute(vfn, [3 1 2]);

  v_tol = abs(max(v_guess(:) - vfn(:)));

  v_guess = vfn; % update value functions
end;
pol_indx = permute(pol_indx, [3 1 2]);
```

C The function file pol_fn_iter.m

```
% policy function iteration by Prof. Kuhn
k = 30;
v_tol = 1;
while v_tol > 1e-6;
  % construct total return function
  v_mat = ret + beta * repmat(permute(PI * v_guess, [3 2 1]), [num_a 1 1]);
  % choose highest value (associated with a' choice)
  [vfn, pol_indx] = max(v_mat, [], 2);
  vfn = permute(vfn, [3 1 2]);
  pol_indx = permute(pol_indx, [3 1 2]);
  v_tol = abs(max(v_guess(:) - vfn(:)));
  v_guess = vfn; % update value functions
  % construct Q matrix from policy index and PI
  Q_mat = makeQmatrix(pol_indx, PI);
  % construc return vector and value vector
  pol_fn = a(pol_indx);
  u_mat = bsxfun(@minus, r * a, pol_fn);
  u_mat = bsxfun(@plus, u_mat, z' * w);
  u_mat = (u_mat .^ (1 - sigma)) ./ (1 - sigma);
  u_vec = u_mat(:);
  w_vec = v_guess(:);
  % PFI
  for ii = 1:k
      w_vec_new = u_vec + beta * Q_mat * w_vec;
      w_vec = w_vec_new;
  end
  v_guess = reshape(w_vec, num_z, num_a);
end;
```