1 main.m

```
% Solving an Aiyagari model economy
% by Espen Henriksen

clear all;   % clears all variables from the memory
close all;   % closes all figures
tic  % starts the "stop watch"

alpha = .35;
delta = .025;
gamma = 5;
beta    =  .98;
sigma   = 2;

Pi = [.98  .02
      .40  .60];

dim = 2;      % Number of values the exogenous state variable might take
ag  = 51;     % Number of grid points for both the control var and state var k
alow  = 220;
ahigh = 450;
agrid = linspace(alow,ahigh,ag)';

param = [Pi(:,1)' Pi(:,2)' alpha delta gamma beta sigma dim ag alow ahigh]';

%kstar  = bisection('aiyagari',100,600,param);
kstar = newton('aiyagari',100,param);

[k1,dist] = aiyagari(kstar,param);

plot(agrid,dist);

tmp = Pi^10000;
tmp = tmp(2,2);
figure;plot(agrid,dist(:,2)/tmp)
print -depsc2 dist_cond_u.eps
```

2 bisection.m

```
function kstar = bisection(FUN,a,b,param);

[k1 dist] = feval(FUN,a,param);
fa = a - k1;
fprintf('Bisection: \t a = %3.8f \t fa = %3.8f \n', [a fa]);

[k1 dist] = feval(FUN,b,param);
fb = b - k1;
fprintf('Bisection: \t b = %3.8f \t fb = %3.8f \n', [b fb]);

diff = 1;
convcrit = 10^-5;
```

```
while diff > convcrit
    c = (a+b)/2;
    [k1 dist] = feval(FUN,c,param);
    fc = c - k1;
    fprintf('Bisection: \t c = %3.8f \t fc = %3.8f \n', [c fc]);
    if fc*fa < 0
        b  = c;
        fb = fc;
    else
        a  = c;
        fa = fc;
    end
    diff = abs(fc);
end

return
```

3 newton.m

```
function kstar = newton(FUN,k0,param);

h = 10^-7;

fprintf('Newton: \t k0 = %3.8f  \n', [k0]);

diff = 1;
convcrit = 10^-6;
while diff > convcrit
    fk0  = k0 - feval(FUN,k0,param);
    fk0plush = (k0+h) - feval(FUN,k0+h,param);
    fk0minh  = (k0-h) - feval(FUN,k0-h,param);
    dfk0 = 1/(2*h)*(fk0plush - fk0minh);

    k1 = k0 - fk0/dfk0;

    diff = abs(k0 - k1);
    fprintf('Newton: \t k1 = %3.8f \t diff = %3.8f \n', [k1 diff]);
    k0 = k1;
end

kstar = k1;

return
```

4 aiyagari.m

```
function [k1,dist] = aiyagari(k0,param)

    Pi(:,1) = param(1:2,1);
    Pi(:,2) = param(3:4,1);
```

```
alpha = param(5);
delta = param(6);
gamma = param(7);
beta  = param(8);
sigma = param(9);
dim   = param(10);
ag    = param(11);
alow  = param(12);
ahigh = param(13);

N = Pi^10000;
N = N(1,1);

% Compute prices -- ie. wage and rental rates
r = alpha     *gamma*(k0/N)^(alpha-1) - delta;
w = (1-alpha)*gamma*(k0/N)^(alpha) ;

agrid = linspace(alow,ahigh,ag)';

c = zeros(ag,ag,dim);
u = zeros(ag,ag,dim);

% i is a counter for the exog. state variable e,u
% j is a counter for the endogenous state variable a
% m is a counter for the control variable a'

warning off; % disable warning for taking log of zero (just annoying)
for i = 1 : dim
    for j = 1 : ag
        for m = 1 : ag
            if i == 1
                c(j,m,i) = (1+r)*agrid(j) + w - agrid(m);
            elseif i == 2
                c(j,m,i) = (1+r)*agrid(j)     - agrid(m);
            end
            if c(j,m,i) < 0
                c(j,m,i) = 0;
            end
            if sigma == 1
                u(j,m,i) = log(c(j,m,i));
            else
                u(j,m,i) = ((c(j,m,i))^(1-sigma)-1)/(1-sigma);
            end
        end
    end
end
warning on;   % turn warnings on again
clear c       % free up memory
clear i j m n % clean up

v  = zeros(ag,dim);
Tv = zeros(ag,dim);

convcrit = 1E-11;   % chosen convergence criterion
```

```
diff = 1;           % arbitrary initial value greater than convcrit
iter = 0;           % iterations counter

while diff > convcrit
    diff = 0;
    for i = 1 : dim
        objfn(:,:,i) = u(:,:,i) + beta*(Pi(i,1)*((v(:,1)*ones(1,ag))')) + ...
                                  (Pi(i,2)*((v(:,2)*ones(1,ag))'));
        Tv(:,i) = max(objfn(:,:,i),[],2);
    end
    diff = norm(v-Tv);
    v = Tv;
    iter = iter + 1;
end
fprintf('␣Value␣fn␣iterations:␣%5.0f',iter);

for i = 1 : dim
    objfn(:,:,i)    = u(:,:,i) + beta*(Pi(i,1)*((v(:,1)*ones(1,ag))'))+...
                                 (Pi(i,2)*((v(:,2)*ones(1,ag))'));
    [tmp1,x1]       = max(objfn(:,:,i),[],2);
    agridrule(:,i) = x1;
    adecrule(:,i)  = agrid(agridrule(:,i));
    if i == 1
        cdecrule(:,1)  = (1+r)*agrid + w - adecrule(:,1);
    elseif i ==2
        cdecrule(:,2)  = (1+r)*agrid     - adecrule(:,2);
    end
end

clear iter i objfn v Tv x1 tmp1

% Uniform initial distribution
dist = ones(ag,dim)/(2*ag);

diff = 1;
iter = 0;

while diff > convcrit
    ndist = zeros(ag,dim);
    %
    for i = 1 : dim
        for j = 1 : ag
            %
            for p = 1 : dim
                for q = 1 : ag
                    if agridrule(q,p) == j
                        ndist(j,i) = ndist(j,i) + Pi(p,i)*dist(q,p);
                    end
                end
            end
        end
    end
    diff = norm(dist-ndist);
    dist = ndist;
```

```
        iter = iter + 1;
    end

    fprintf('\t Iterations on the dist: %5.0f',iter);

    k1 = agrid'*dist(:,1) + agrid'*dist(:,2);
    newr = alpha     *gamma*(k1/N)^(alpha-1) - delta;
    neww = (1-alpha)*gamma*(k1/N)^(alpha) ;
    r = newr;
    w = neww;
    fprintf(' \t r = %1.6f \t w = %2.6f \n',[r w]);

return
```

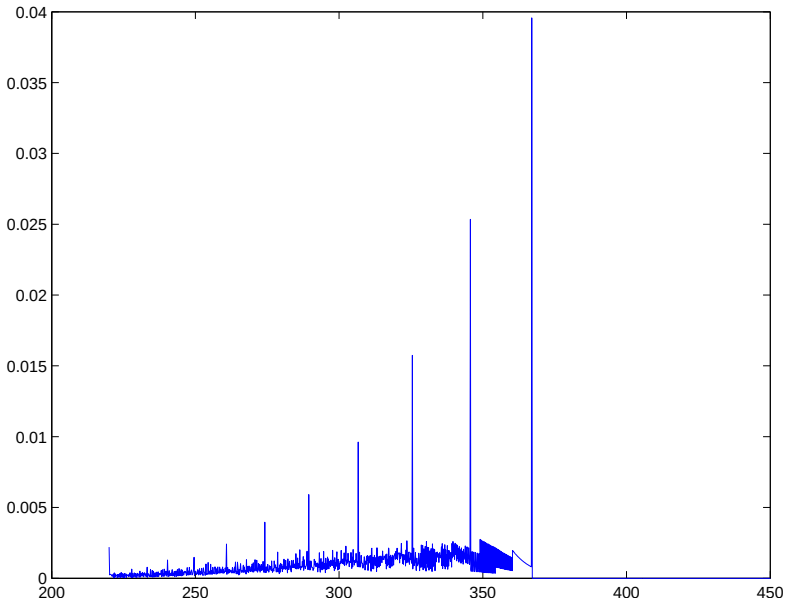

Figure 1: Asset distribution conditional on being unemployed, 1551 grid points