```
% ECON 5300, UiO, Fall 2015
% this file solves Ex 3.3 of Problem Set 3
% note: - this script calls the user written function equilibrium.m
%       - uses the Chebyshev collocation method

%% 1. housekeeping

clc;                % clear the command window

%% 2. define parameters and functional forms

% parameters
alpha  = .36;       % capital income share
beta   = [illegible]; % subjective discount factor
delta  = .10;       % depreciation rate
sigma  = 2;         % relative risk aversion parameter

% functional forms
% note: - define utility, production function, and rental rate directly in
%       this file. these are so-called anonymous functions (they are not
%       located in a separate .m file).
%       - uses element-wise operations, to be compatible with matrices and
%       vectors

% utility function
if sigma ~= 1
    u  = @(c) (c.^(1-sigma)-1)/(1-sigma);
else
    u = log(c);     % approaches log when sigma=1
end

% marginal utility
u1 = @(c) c.^(-sigma);

% production function
f  = @(k) k.^alpha;

% rental rate
f1 = @(k) [illegible]

%% 3. compute steady-state

% physical capital
kstar = (alpha*beta/(1-beta*(1-delta)))^(1/(1-alpha));

% consumption
% note: here you can make use of the previously declared anonymous
%       functions, here f(k).
cstar = f(kstar)+(1-delt[illegible]

% value
Vstar = u(cstar)/(1-beta);

%% 4. set up the function space using the function approximation toolkit
```

```
% dimension
n = 6;                    % number of collocation points
d = 3;                    % number of unknown functions: g(k),c(k),V(k)

% set bounds for the physical capital grid around the steady-state
kmin = 0.5*kstar;
kmax = 1.5*kstar;

% use Chebyshev basis functions
fspace = fundefn('cheb',n,kmin,[redacted];
k      = gridmake(funnode(fspace));

%% 5. initial guess for the unknown functions

% guess derivative of the policy functions with respect to k
c1 = [redacted];
g1 = .9;
V1 = 1;

% linear guess around the steady-state for the policy functions
c = cstar+c1*(k-kstar);
g = kstar+g1*(k-kstar);
V = [redacted]

% translate the above guess into coefficients using the function
% approximation toolkit
% note: coefficients in the first column, cf(:,1), will be the ones
%       associated to c, cf(:,2), to g, and cf(:,3) to V. The dimensionality
%       of the coefficient matrix cf is therefore (n,d).
cf = funfitxy([redacted],k,[c,g,V]);

%% 6. solve for the dynamic equilibrium with collocation

% prepare the root-finding routine
optset('broyden','showiters','true');       % display iterations and the residual
fprintf('Solving for the dynamic equilibrium:\n')

% use CompEcon's root-finding routine broyden to find the "root" of the
% external function equilibrium.m which evaluates the equilibrium
% conditions given the coefficients in cf. Finding the "root" or the "zero"
% is equivalent to setting the stated equilibrium conditions equal to zero.
% note: cf(:) stacks the coefficient matrix into a single column vector.
%       Pass the necessary objects, parameters, and functions to the
%       routine as well.
cf = broyden('equilibrium',cf(:),k,fspace,n,d,beta,delta,u,u1,f,f1);

% unstack coefficients
cf = reshape(cf,n,d);

fprintf('Done.\n')

%% 7. plot the policy functions

% set up finer evaluation grid
N     = 100;                        % number of grid points
```

```
kplot = linspace(kmin,kmax,N)';   % equally spaced (column) grid for k

% evaluate functions at the finer grid
cplot = funeval(cf(:,1),fspace,kplot);
gplot = funeval(cf(:,2),fspace,kplot);
Vplot = funeval(cf(:,3),fspace,kplot);

% set up figure
scrsz = get(0,'ScreenSize');
figure('Position',[scrsz(3)*1/4 scrsz(4)*1/4 scrsz(3)*1/2 ...
    scrsz(4)*1/2]);

% consumption function
subplot(2,2,1)                          % first subplot in a 2-by-2 figure
plot(kplot,cplot)                       % plot cplot against kplot
title('Consumption function');          % set title of the subfigure
xlim([kmin,kmax]);                      % set limits of the horizontal-axis
xlabel('Capital, k'); ylabel('c(k)');   % label the axes

% savings function
subplot(2,2,2)
plot(kplot,[gplot,kplot])
title('Savings function');
xlim([kmin,kmax]);
xlabel('Capital, k'); ylabel('g(k)');

% value function
subplot(2,2,3)
plot(kplot,Vplot)
title('Value function');
xlim([kmin,kmax]);
xlabel('Capital, k'); ylabel('V(k)');
```

```
function res = equilibrium(cf,k,fspace,n,d,beta,delta,u,u1,f,f1)
%EQUILIBRIUM evaluates the residual of the dynamic equilibrium conditions
%for given coefficients

% unstack coefficients
cf = reshape(cf,n,d);

% evaluate equilibrium functions
x = funeval(cf,fspace,k);
c = x(:,1);
g = x(:,2);
V = x(:,3);

% evaluate future consumption and value (as a function of future capital, g(k))
cp = funeval(cf(:,1),fspace,g);
Vp = funeval(cf(:,3),fspace,g);

% Bellman equation
res(:,1) = u(c)+beta*Vp-V;

% Euler equation
res(:,2) = u1(c)-beta*(f1(g)+(1-delta)).*u1(cp);

% resource constraint
res(:,3) = f(k)+[redacted]

% stack residuals and return
res = res(:);

end
```