ECON 6140 - Problem Set # 3

Julien Manuel Neves

February 15, 2018

Open Economy with Durable Goods

(1) The Langrangian of the problem:

$$\begin{aligned}\mathcal{L} = \sum_{t=0}^{\infty} \beta^t [u(c_t) + v(z_{t+1}) + \lambda_t (R^* b_t + af(k_t) - c_t - x_{kt} - q x_{zt} - b_{t+1} - \frac{d}{2}(z_{t+1} - z_t)^2) \\ + \mu_t (x_{zt} + (1-\delta_z) z_t - z_{t+1}) \\ + \nu_t (x_{kt} + (1-\delta_k) k_t - k_{t+1})]\end{aligned}$$

KKT:

$$\begin{aligned}
c_t &: c_t^{-\eta} - \lambda_t = 0 \\
z_{t+1} &: z_{t+1}^{-\eta} - \lambda_t d(z_{t+1} - z_t) + \beta \lambda_{t+1} d(z_{t+2} - z_{t+1}) - \mu_t + \beta \mu_{t+1}(1-\delta_z) = 0 \\
b_{t+1} &: \beta \lambda_{t+1} R^* - \lambda_t = 0 \\
k_{t+1} &: \beta \lambda_{t+1} a f'(k_{t+1}) + \beta \nu_{t+1}(1-\delta_k) - \nu_t = 0 \\
x_{zt} &: \mu_t - q\lambda_t = 0 \\
x_{kt} &: \nu_t - \lambda_t = 0 \\
&: R^* b_t + af(k_t) \geq c_t + x_{kt} + q x_{zt} + b_{t+1} + \frac{d}{2}(z_{t+1} - z_t)^2 \\
&: x_{zt} + (1-\delta_z) z_t \geq z_{t+1} \\
&: x_{kt} + (1-\delta_k) k_t \geq k_{t+1} \\
TVC_1 &: \lim_{T\to\infty} \beta^T u_c(c_T) k_{T+1} = 0 \\
TVC_2 &: \lim_{T\to\infty} \beta^T u_c(c_T) b_{T+1} = 0
\end{aligned}$$

Note that we need an extra tranversality condition for capital.

Note that since $R^* = \frac{1}{\beta}$, then $\lambda_t = \lambda_{t+1}$.

The steady state FOCs are given by

$$\left(\frac{c}{z}\right)^{\eta} = q(1 - \beta(1-\delta_z))$$
$$\beta(af'(k) + (1-\delta_k)) = 1$$

and the steady state constraint yield

$$\begin{aligned}(R^*-1)b + af(k) &= c + x_k + qx_z \\ x_z &= \delta_z z \\ x_k &= \delta_z k\end{aligned}$$

Note that if $f'(\cdot)$ is decreasing, k is uniquely determined by $(\beta, a, \delta_k, f(\cdot))$.

Since d doesn't enter these equations, hence it is irrelevant for the steady states.

If $q \uparrow$ or $\delta_z \uparrow$, then $\frac{c}{z} \uparrow$. Since k is only determined by the coefficient, we need b to adjust such that $(R^*-1)b + af(k) = c + q\delta_z z + \delta_z k$ holds for the new steady state.

(2) Recall that since $R^* = \frac{1}{\beta}$, we have $\lambda_t = \lambda_{t+1} = \lambda$. This in turns implies that $\mu_t = q\lambda$ and $\nu_t = \lambda$, i.e. every Lagrange multiplier is constant. This implies that c_t and z_{t+1} are also constant for $t \geq 0$. Note that z_0 is given, therefore $z_t = z$ for $t \geq 1$ which might not be equal to z_0. In fact, z is equal to the steady state value of

$$z = c[q(1-\beta(1-\delta_z))]^{-\frac{1}{\eta}}$$

Moreover, we get

$$\beta(af'(k_{t+1}) + (1-\delta_k)) = 1$$

i.e. k_{t+1} is also constant if $f'(\cdot)$ is decreasing and k is given by $\beta(af'(k) + (1-\delta_k)) = 1$.

Since k_{t+1} is constant $x_{kt} + (1-\delta_k)k_t = k_{t+1}$ becomes $x_{kt} = \delta_k k$ for $t \geq 1$ and $x_{k0} = k - (1-\delta_k)k_0$ for $t = 0$. Since z_{t+1} is constant $x_{zt} + (1-\delta_z)z_t = z_{t+1}$ becomes $x_{zt} = \delta_z z$ for $t \geq 1$ and $x_{z0} = z - (1-\delta_z)z_0$ for $t = 0$.

Now, we look at the dynamics of b_t. Note that it is fully determined by

$$b_{t+1} = R^* b_t + af(k_t) - c - x_{kt} - qx_{zt}$$

Since we know b_0, the whole sequence of x_{kt} and x_{zt}, the only thing missing is c. As shown previously, c_t is constant. Hence, to pin down b_t we need to find $c_0 = c$.

Let's look at the phase diagram of b and z with $t \geq 1$. Note that we get the following equation

$$\begin{aligned}z_{t+1} &= z_t = z \\ b_{t+1} &= R^* b_t + af(k) - c - \delta_k k - q\delta z_t\end{aligned}$$

Hence, the locus for b is

$$b_t = -\frac{a\beta}{1-\beta}f(k) + \frac{c\beta}{1-\beta} + \frac{\delta_k\beta}{1-\beta}k + \frac{q\delta_z\beta}{1-\beta}z_t$$

Hence, any point that satisfy $b_t = -\frac{a\beta}{1-\beta}f(k) + \frac{c\beta}{1-\beta} + \frac{\delta_k\beta}{1-\beta}k + \frac{q\delta_z\beta}{1-\beta}z$ is the bond steady state. To get on this locus, we need b_0 to be such that $b_1 = -\frac{a\beta}{1-\beta}f(k) + \frac{c\beta}{1-\beta} + \frac{\delta_k\beta}{1-\beta}k + \frac{q\delta_z\beta}{1-\beta}z$.

Hence, c_0 needs to be such that $b_1 = R^* b_0 + af(k_0) - c_0 - x_{k0} - qx_{z0}$ holds and voilà!

(3) Again, every Lagrange multiplier are constant. Therefore, we get

$$\beta(af'(k_{t+1}) + (1 - \delta_k)) = 1$$

i.e. k_{t+1} is constant if $f'(\cdot)$ is decreasing. Since k_{t+1} is constant $x_{kt} + (1 - \delta_k)k_t = k_{t+1}$ becomes $x_{kt} = \delta_k k$ for $t \geq 1$ and $x_{k0} = k - (1 - \delta_k)k_0$ for $t = 0$.

Moreover, the FOC of z_{t+1} is now

$$z_{t+1}^{-\eta} = \lambda[d(z_{t+1} - z_t) - \beta d(z_{t+2} - z_{t+1}) + q(1 - \beta(1 - \delta_z))]$$

Since capital is constant, we can focus on a phase diagram to see what happens to z_t only. Let $y_t = z_{t+1} - z_t$. Hence, we have the two following dynamic equations

$$\begin{aligned} z_{t+1} &= y_t + z_t \\ y_{t+1} &= \frac{1}{\beta} y_t + \frac{q}{\beta d}(1 - \beta(1 - \delta_z)) - \frac{1}{\lambda \beta d}(y_t + z_t)^{-\eta} \end{aligned}$$

Take the loci where z is constant. This yields $y = 0$, i.e. the y-axis.

For the second loci is shape is not too important for our analysis. We simply note that for $y = 0$, we get

$$z^{-\eta} = \lambda q(1 - \beta(1 - \delta_z))$$

Since, $\lambda = c^{-\eta}$, we have $z^{-\eta} = c^{-\eta} q(1 - \beta(1 - \delta_z))$, i.e. the steady state derived previously.

To describe the dynamics, take $y_t < 0$. This implies that $z_{t+1} < z_t$. Therefore on the steady path, if we start over the steady state, z_t will decrease until it reaches z. The reverse can be said for $y_t > 0$. The speed of converge is describe in part 4).

(4) Note that as derived previously, capital goods are constant for period $t \geq 1$. Hence to converge to the steady state, we only one period.

For durable goods, our analysis implies that regardless if we start over or under the second locus, the y_t will converge to the steady state $y = 0$ over time. This in turns implies that the difference between z_{t+1} and z_t will decrease every period until $z_t = z$. Hence, the convergence of durable goods is not "instantaneous" like capital.

Transition paths in the one sector growth model

(1) With the parameters defined in the problem set, we get that the steady state is $k^* = 10.03$ and $c^* = 1.639$. Thus, our starting value is $k_0 = 9.027599$. Figure 1 shows the transition of k_0 on the steady path. It takes 53 periods to converge to the steady state when starting from $k_0 = .9k^*$.

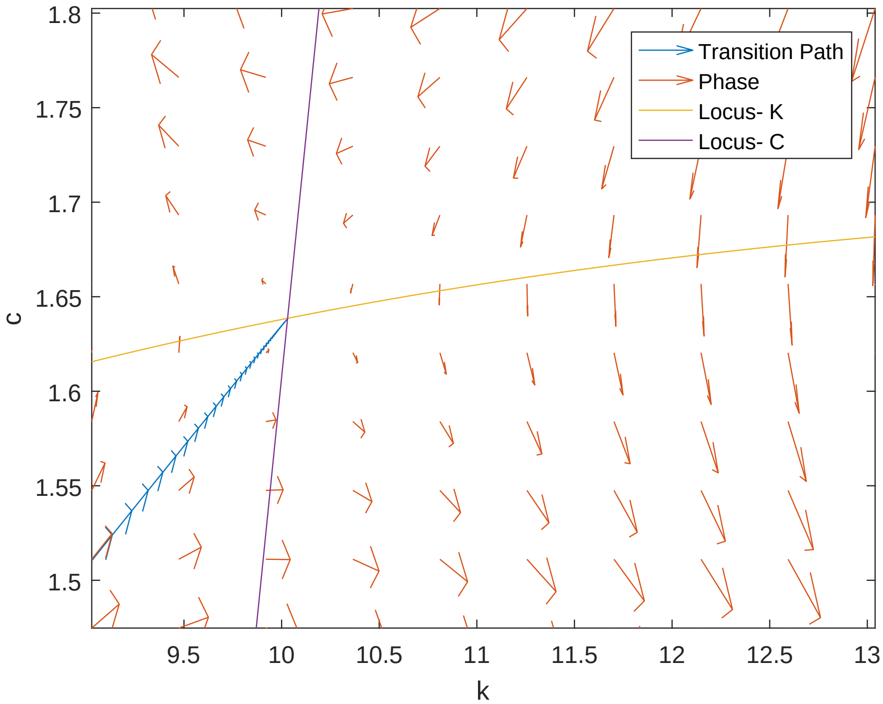

Figure 1: Transition path for growth model starting at $k_0 = .9k^*$

Note that we also plot the loci of our phase diagram on Figure 1. Recall that they are given by

$$
\begin{aligned}
k: & c_t = Ak_t^\alpha - \delta k_t \\
c: & c_t = k_t^\alpha + (1-\delta)k_t - k^*
\end{aligned}
$$

where k^* is the steady state value equal to $\left(\frac{\alpha\beta A}{1-\beta+\beta\delta}\right)^{\frac{1}{1-\alpha}}$.

(2) See Figure 2.

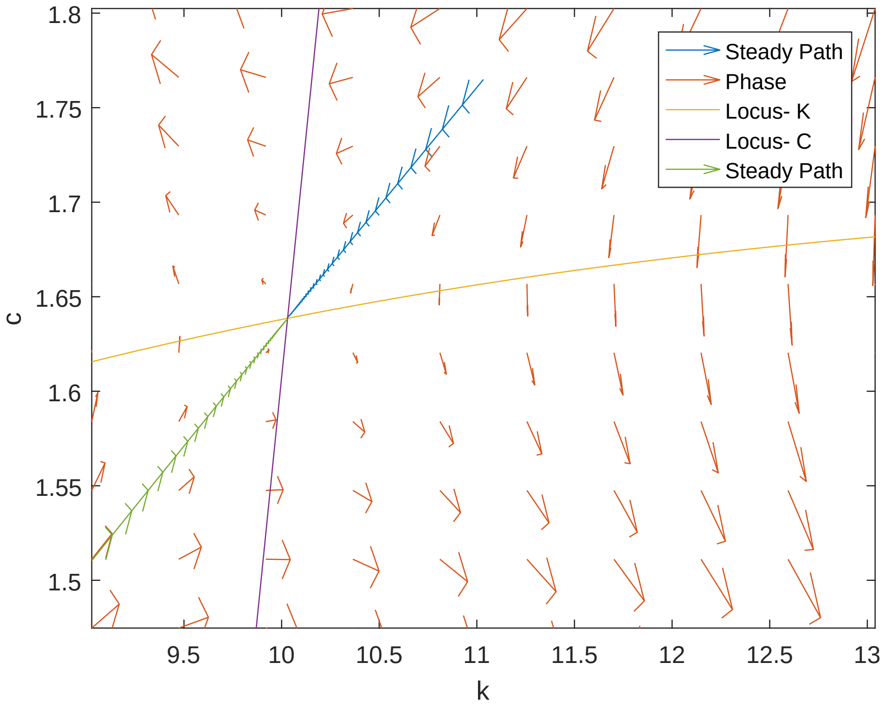

Figure 2: Phase diagram for growth model

Note that the scales on this figure are not 1-1.

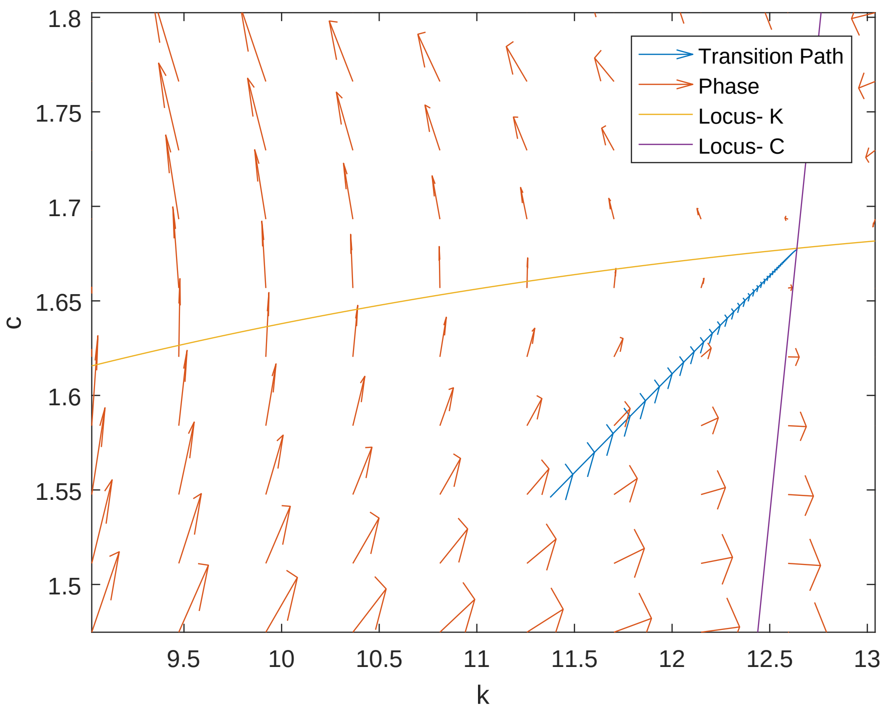

Figure 3: Phase diagram for growth model with increase of 1% in β

(3) By increasing β by 1% the second locus shift to the right and we get that the new steady state is $k^* = 12.640$ and $c^* = 1.678$. Therefore the starting value is $k_0 = 11.376$. Note that since k^* is bigger, k_0 is a bit farther in absolute terms. This results in a longer convergence term of 59 periods. The transition path is plotted in Figure 3.

(4) Note that if A goes up, the first locus also goes up. This implies that the new steady path also shifts up and the new steady state will be one where $k^* \uparrow$ and $c^* \uparrow$.

In this problem, we let the increase in A happen at $t = 1$. Moreover, let's assume that it was an unexpected increase permanent increase in A, i.e. consumer don't preventively adjust to a future steady path.

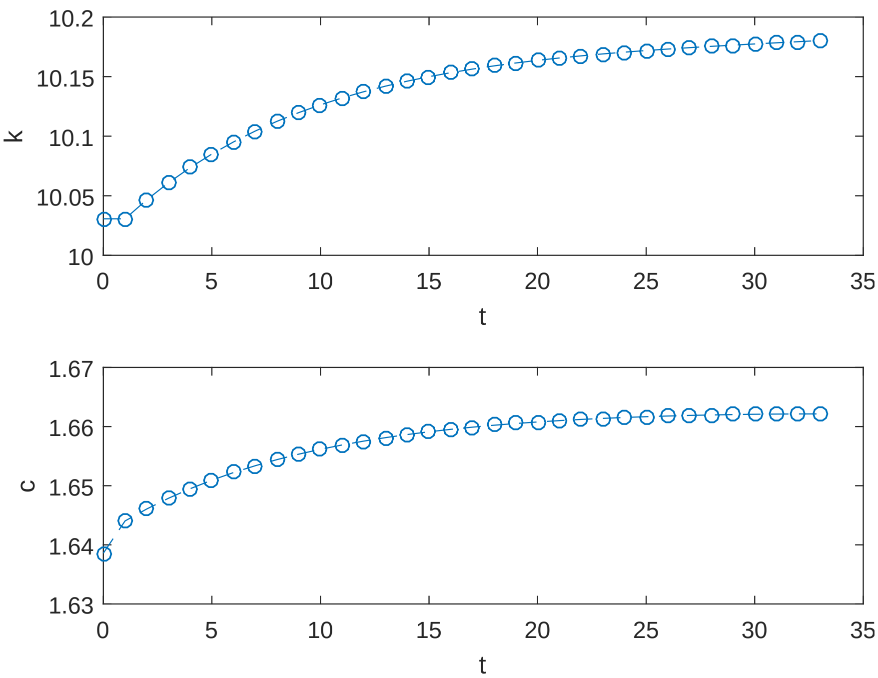

Figure 4: Time path for consumption and capital with increase of 1% in A

Now, since the point (k_0, c_0) is the old steady , it might not be on the new steady path. To adjust to that the consumer will instantaneously change it consumption at time $t = 1$ to get on the new steady path unlike k_0 that can't act reactively. From there on on out, both k_t and c_t will monotonically increase until it reaches the new steady path as shown in Figure 4.

Code

```
%% Shooting Algorithm
% Course: ECON 6140
% Version: 1.0
% Author: Julien Neves
```

```
%% Question
tol = .001; % tolerance
N = 100000; % grid size
T = 600;    % time periods

alpha = 0.33;   % labor share
delta = 0.05;   % depreciation of capital
sigma = 0.5;    % CRRA
beta = .98;     % discount factor
A = 1;          % technology

k = zeros(1,T+1);   % initial k path vector
c = zeros(1,T+1);   % initial c path vector

kstar = ((alpha*beta*A)/(1-beta+beta*delta))^(1/(1-alpha)); % k steady state
cstar =  A*kstar^alpha+(1-delta)*kstar-kstar;   % c steady state

k0 = 0.9*kstar; % starting k value

lb_k = 0.9*kstar;   % lower bound of k axis
ub_k = 1.3*kstar;   % upper bound of k axis

lb_c = 0.9*cstar;   % lower bound of c axis
ub_c = 1.1*cstar;   % lower bound of c axis

axis_k = lb_k : (ub_k-lb_k)/(N-1) : ub_k;   % k axis
axis_c = lb_c : (ub_c-lb_c)/(N-1) : ub_c;   % c axis

crit = 1;   % initialize tolerance criteria
ite = 1;    % initialize iteration

while (crit>tol && ite<=length(axis_c))
    k(1) = k0;  % set starting k0
    c(1) = axis_c(ite); % pick c0
    for t = 1:T
        k(t+1) = A*k(t)^alpha+(1-delta)*k(t)-c(t); % compute k(t+1)
        c(t+1) = c(t)*(beta*alpha*A*k(t+1)^(alpha-1)+beta*(1-delta))^(1/sigma); %compute c(t+1)
        crit = max(abs(kstar-k(t+1)),abs(cstar-c(t+1)));     % deviation from steady state
        if crit<=tol
            % if close to steady state stop algorithm
```

```
                k = k(1:t+1); % cut path after convergences
                c = c(1:t+1); % cut path after convergences
                break
            else
                continue
            end
        end
        ite = ite + 1;  % update iteration
end

u = gradient(k);    % compute k gradient of steady path
v = gradient(c);    % compute c gradient of steady path

loci_k = A*axis_k.^alpha - delta*axis_k;    % compute k loci
loci_c = axis_k.^alpha +(1-delta)*axis_k-kstar; % compute c loci

[K,C]= meshgrid(lb_k : (ub_k-lb_k)/(10-1) : ub_k,lb_c : (ub_c-lb_c)/(10-1) : ub_c);  % create (k,c) grid

dK = A*K.^alpha-delta.*K-C; % compute k gradient of every point in grid
dC = C.*(beta*alpha*A*(dK+K).^(alpha-1)+beta*(1-delta)).^(1/sigma)-C;  % compute c gradient of every point in grid

% plot steady path, phase diagram, and locus
figure(1)
quiver(k,c,u,v,0)
axis([lb_k ub_k lb_c ub_c])
xlabel('k') % x-axis label
ylabel('c') % y-axis label
hold on
quiver(K,C,dK,dC,0)
plot(axis_k,loci_k)
plot(axis_k,loci_c)
legend('Transition Path', 'Phase','Locus- K','Locus- C')
hold off
print -depsc fig1.eps

fprintf('Question 1 \nStarting - K : %f \nSteady state - K : %f \nSteady state - C : %f \nConvergence time : %d.\n\n',k0,kstar,cstar,t);

kstar_old = kstar;  % store original k steady state
cstar_old = cstar;  % store original c steady state
k_old = k;  % store transition path for k
```

```
c_old = c;  % store transition path for c
u_old = u;  % store transition path for k
v_old = v;  % store transition path for c

%% Question 2
k = zeros(1,T+1);   % initial k path vector
c = zeros(1,T+1);   % initial c path vector

k0 = 1.1*kstar; % starting k value

crit = 1;   % initialize tolerance criteria
ite = 1;    % initialize iteration

while (crit>tol && ite<=length(axis_c))
    k(1) = k0;  % set starting k0
    c(1) = axis_c(ite); % pick c0
    for t = 1:T
        k(t+1) = A*k(t)^alpha+(1-delta)*k(t)-c(t); % compute k(t+1)
        c(t+1) = c(t)*(beta*alpha*A*k(t+1)^(alpha-1)+beta*(1-delta))^(1/sigma); %compute c(t+1)
        crit = max(abs(kstar-k(t+1)),abs(cstar-c(t+1)));     % deviation from steady state
        if crit<=tol
            % if close to steady state stop algorithm
            k = k(1:t+1); % cut path after convergences
            c = c(1:t+1); % cut path after convergences
            break
        else
            continue
        end
    end
    ite = ite + 1 ; % update iteration
end

u = gradient(k);    % compute k gradient of steady path
v = gradient(c);    % compute c gradient of steady path

% plot steady path, phase diagram, and locus
figure(2)
quiver(k,c,u,v,0)
hold on
quiver(K,C,dK,dC,0)
plot(axis_k,loci_k)
plot(axis_k,loci_c)
```

```
quiver(k_old,c_old,u_old,v_old,0)
hold off
axis([lb_k ub_k lb_c ub_c])
xlabel('k') % x-axis label
ylabel('c') % y-axis label
legend('Steady Path', 'Phase','Locus- K','Locus- C','Steady Path')
print -depsc fig2.eps


%% Question 3
alpha = 0.33;   % labor share
delta = 0.05;   % depreciation of capital
sigma = 0.5;    % CRRA
beta = .9898;     % discount factor
A = 1;          % technology

k = zeros(1,T+1);   % initial k path vector
c = zeros(1,T+1);   % initial c path vector

kstar = ((alpha*beta*A)/(1-beta+beta*delta))^(1/(1-alpha)); % k
    steady state
cstar =  A*kstar^alpha+(1-delta)*kstar-kstar;   % c steady state

k0 = 0.9*kstar; % starting k value

% note that we use the axis define in question 1
axis_k = lb_k : (ub_k-lb_k)/(N-1) : ub_k;   % k axis
axis_c = lb_c : (ub_c-lb_c)/(N-1) : ub_c;   % c axis

crit = 1;   % initialize tolerance criteria
ite = 1;    % initialize iteration

while (crit>tol && ite<=length(axis_c))
    k(1) = k0;  % set starting k0
    c(1) = axis_c(ite); % pick c0
    for t = 1:T
        k(t+1) = A*k(t)^alpha+(1-delta)*k(t)-c(t); % compute k(t
            +1)
        c(t+1) = c(t)*(beta*alpha*A*k(t+1)^(alpha-1)+beta*(1-delta
            ))^(1/sigma); %compute c(t+1)
        crit = max(abs(kstar-k(t+1)),abs(cstar-c(t+1)));    %
            deviation from steady state
        if crit<=tol
            % if close to steady state stop algorithm
            k = k(1:t+1); % cut path after convergences
```

```
            c = c(1:t+1); % cut path after convergences
            break
        else
            continue
        end
    end
    ite = ite + 1 ; % update iteration
end

u = gradient(k);    % compute k gradient of steady path
v = gradient(c);    % compute c gradient of steady path

loci_k = A*axis_k.^alpha - delta*axis_k;    % compute k loci
loci_c = axis_k.^alpha +(1-delta)*axis_k-kstar; % compute c loci

[K,C]= meshgrid(lb_k : (ub_k-lb_k)/(10-1) : ub_k,lb_c : (ub_c-lb_c
    )/(10-1) : ub_c);  % create (k,c) grid

dK = A*K.^alpha-delta.*K-C; % compute k gradient of every point in
     grid
dC = C.*(beta*alpha*A*(dK+K).^(alpha-1)+beta*(1-delta)).^(1/sigma)
    -C;  % compute c gradient of every point in grid

% plot steady path, phase diagram, and locus
figure(3)
quiver(k,c,u,v,0)
axis([lb_k ub_k lb_c ub_c])
xlabel('k') % x-axis label
ylabel('c') % y-axis label
hold on
quiver(K,C,dK,dC,0)
plot(axis_k,loci_k)
plot(axis_k,loci_c)
hold off
legend('Transition Path', 'Phase','Locus- K','Locus- C')
print -depsc fig3.eps

fprintf('Question 3 \nStarting - K : %f \nSteady state - K : %f \
    nSteady state - C : %f \nConvergence time : %d.\n\n',k0,kstar,
    cstar,t);


%% Question 4
alpha = 0.33;   % labor share
delta = 0.05;   % depreciation of capital
```

```
sigma = 0.5;     % CRRA
beta = .98;      % discount factor
A = 1.01;        % technology

k = zeros(1,T+1);    % initial k path vector
c = zeros(1,T+1);    % initial c path vector

kstar = ((alpha*beta*A)/(1-beta+beta*delta))^(1/(1-alpha)); % k steady state
cstar =  A*kstar^alpha+(1-delta)*kstar-kstar;    % c steady state

k0 = kstar_old; % starting k value

lb_k = kstar_old;    % lower bound of k axis
ub_k = 1.1*kstar;    % upper bound of k axis

lb_c = 0.9*cstar;    % lower bound of c axis
ub_c = 1.1*cstar;    % lower bound of c axis

axis_k = lb_k : (ub_k-lb_k)/(N-1) : ub_k;    % k axis
axis_c = lb_c : (ub_c-lb_c)/(N-1) : ub_c;    % c axis

crit = 1;    % initialize tolerance criteria
ite = 1;     % initialize iteration

while (crit>tol && ite<=length(axis_c))
    k(1) = k0;  % set starting k0
    c(1) = axis_c(ite); % pick c0
    for t = 1:T
        k(t+1) = A*k(t)^alpha+(1-delta)*k(t)-c(t); % compute k(t+1)
        c(t+1) = c(t)*(beta*alpha*A*k(t+1)^(alpha-1)+beta*(1-delta))^(1/sigma); %compute c(t+1)
         crit = max(abs(kstar-k(t+1)),abs(cstar-c(t+1)));     % deviation from steady state
         if crit<=tol
             % if close to steady state stop algorithm
             k = k(1:t+1); % cut path after convergences
             c = c(1:t+1); % cut path after convergences
             break
         else
             continue
        end
    end
    ite = ite + 1 ; % update iteration
```

```
end

k = [kstar_old , k]; % add k0
c = [cstar_old , c]; % add c0

% plot time path of k and c
figure(4)
subplot(2,1,1)
plot(0:length(k)-1,k, '--o');
xlabel('t') % x-axis label
ylabel('k') % y-axis label
subplot(2,1,2)
plot(0:length(c)-1,c, '--o');
xlabel('t') % x-axis label
ylabel('c') % y-axis label
print -depsc fig4.eps

fprintf('Question 4 \nStarting - K : %f \nSteady state - K : %f \
    nSteady state - C : %f \nConvergence time : %d.\n\n',k0,kstar,
    cstar,t);
```