```
%This program will solve for the transition path of capital
%from the initial level (k0) to the steady state level (kss)
%for the standard growth model. There is no endogenous labor supply decision.
%The production function is Cobb-Douglas, with a capital share of alpha.
%The utility function is (c^(1-sigma))/(1-sigma).
%The program assumes that the initial level of capital is below the steady
%state value. Specifically, it is assumed that k0=lambda*kss.
%T is the number of periods for which the path is computed.

%Parameter values
delta=.08;
alpha= 1/3;
beta=.96;
lambda=.5;
T=200;
A=1;

sigma_li = [.01, 1.01, 2, 3, 4, 10];
ksol_sigma = zeros(T, 6); %6 sigma values
csol_sigma = zeros(T, 6);


kss=((1/beta-(1-delta))/A/alpha)^(1/(alpha-1));
k0 = lambda*kss;

for j = 1:6
    sigma = sigma_li(j);
    %Solution for path ksol(t)
    ksol(1)=k0;
    ksol_sigma(1, j) = ksol(1);

    for t = 2:T
    %compute path for various choices of k(t), given ksol(t-1)
    %each guess for k(t) will generate a series that has first element
    %ksol(t-1), second element the guess and then subsequent elements given by
    %iterating on the first order conditions. We call this sequence kguess.
        kguess(1) = ksol(t-1);

    %Step 1:  Set boundaries for interval that ksol(t) must lie in.
        kmin = ksol(t-1);
        kmax = kss;

    %Step 2: Pick a point in the interval as a hypothetical choice for k(t)
        while abs(kmax-kmin) > .00000015*kss
            kn=.5*(kmin+kmax);
            kguess(2)=kn;

    %Step 3: Determine the rest of the path given the choice for k(t)
    %stop is simply an indicator to tell us if we need to continue to the next
    %element of the kguess series. i is the index of the kguess series, with
i=1 corresponding
    %to k(t-1) and i=2 corresponding to k(t).
```

```
            stop=0;
            i=2;

            while stop < 1
                i = i+1;
    %implied value for kguess(i)
                kguess(i)=A*kguess(i-1)^alpha+(1-delta)*kguess(i-1)-...
                (beta*(A*alpha*kguess(i-1)^(alpha-1)+(1-delta)))^(1/sigma)*...
                (A*kguess(i-2)^alpha+(1-delta)*kguess(i-2)-kguess(i-1));
    %check to see if kguess is going to zero (i.e., did we enter region I)
                if kguess(i)<=kguess(i-1),
kmin=kn;stop=1;else,kguess(i)=kguess(i);end
    %check to see if kguess is going beyond kss (i.e., did we enter region
    %III)
                if kguess(i)>kss, kmax=kn;stop=1;else,kguess(i)=kguess(i);end
            end
        end
    % We have now determined (within the limits of our approximation) the
next value of k(t)
        ksol(t)=kguess(2);
        ksol_sigma(t, j) = ksol(t);
    end
end

%We now know the whole sequence ksol(t). We can also determine the
%sequences for consumption and utility;
csol_sigma(1:(T-1), :) = A * ksol_sigma(1:(T-1), :).^alpha...
            + (1-delta) * ksol_sigma(1:(T-1), :)...
            - ksol_sigma(2:T, :);

csol_sigma(T, :) = csol_sigma(T-1,:);

uv = (csol_sigma.^(1-sigma))/(1-sigma);

betav(1) = 1;

for j = 2:T
    betav(j) = beta * betav(j-1);
end

utot = betav*uv;

srate = 1 - csol_sigma./(A*ksol_sigma.^alpha);
```

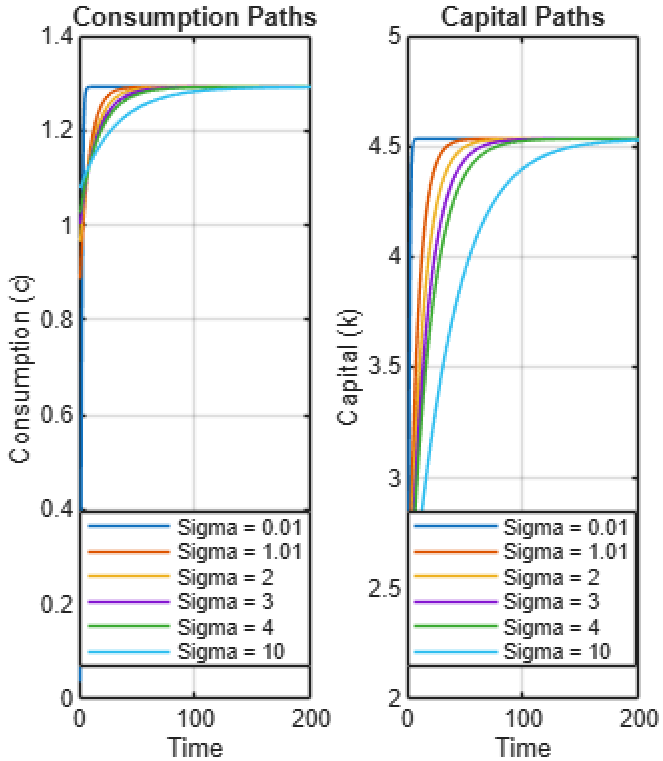
plot

```
clear legend_labels
for i = 1:6
    legend_labels{i} = ['Sigma = ', num2str(sigma_li(i))];
end

figure;
```

```
subplot(1,2,1)
plot(1:T, csol_sigma); % plot consumption paths
title('Consumption Paths');
xlabel('Time');
ylabel('Consumption (c)');
legend(legend_labels, 'Location', 'best');
grid on;

subplot(1, 2, 2); % subplots2
plot(1:T, ksol_sigma); % plot capital paths
title('Capital Paths');
xlabel('Time');
ylabel('Capital (k)');
legend(legend_labels, 'Location', 'best');
grid on;
```



Published with MATLAB® R2025b