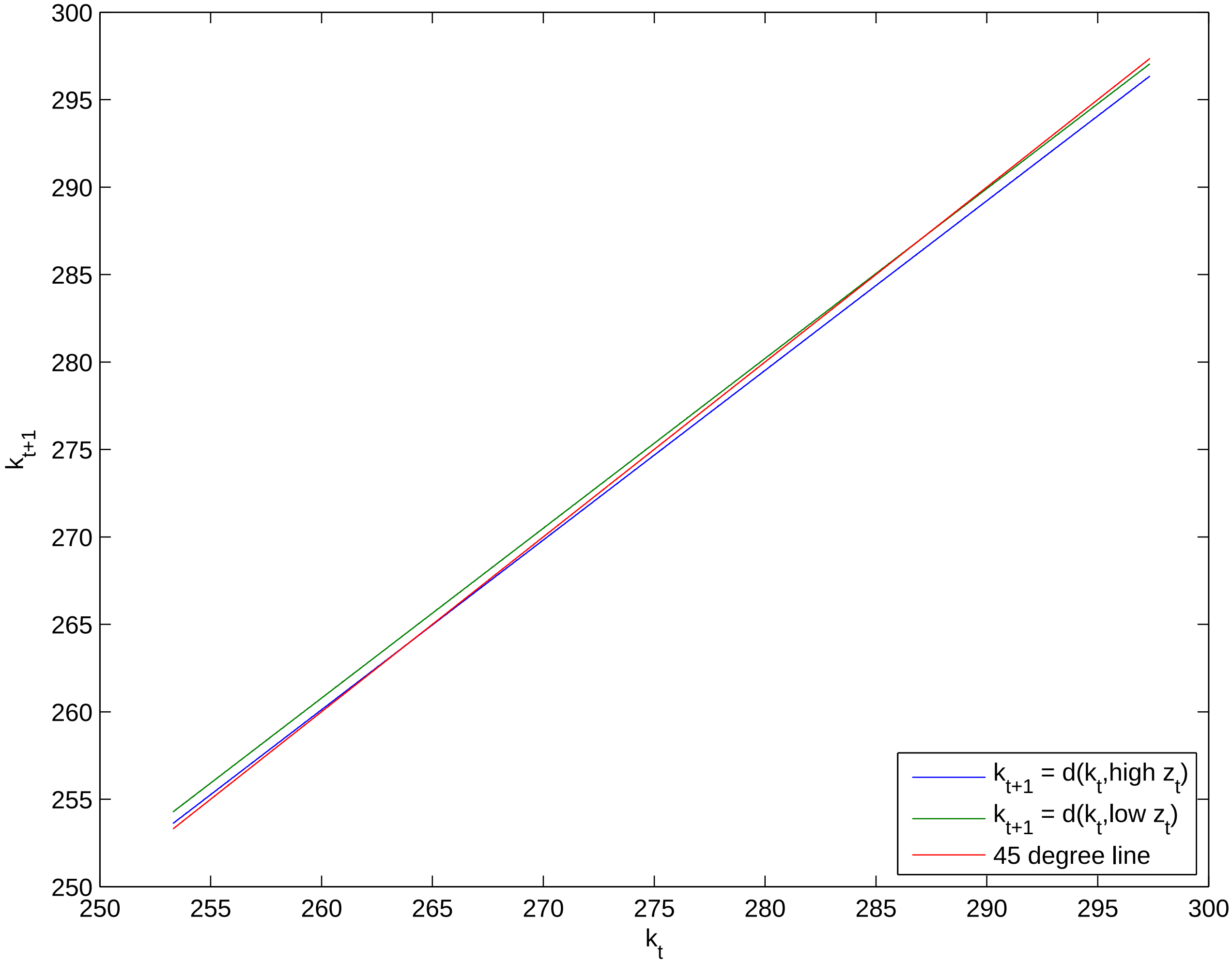
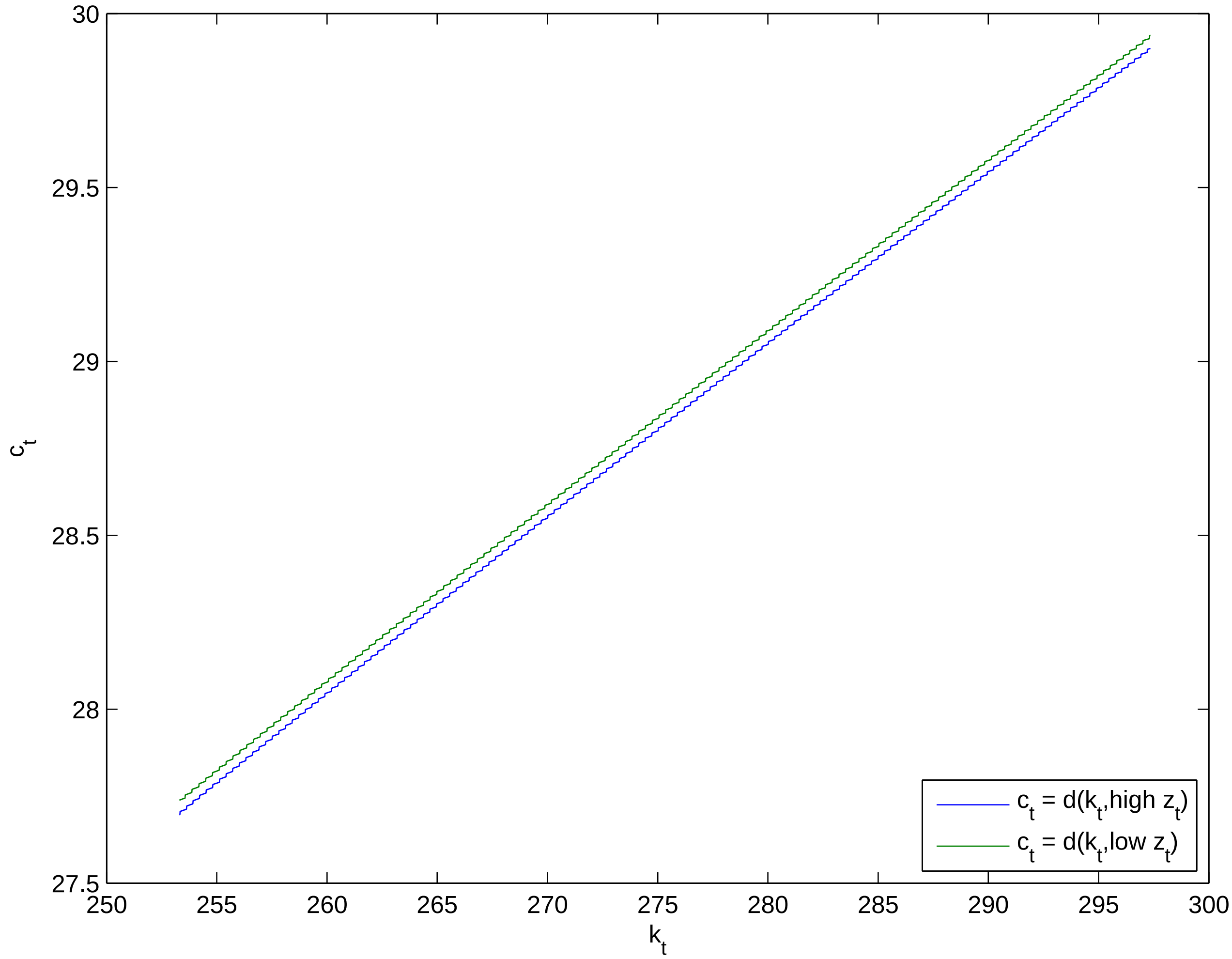
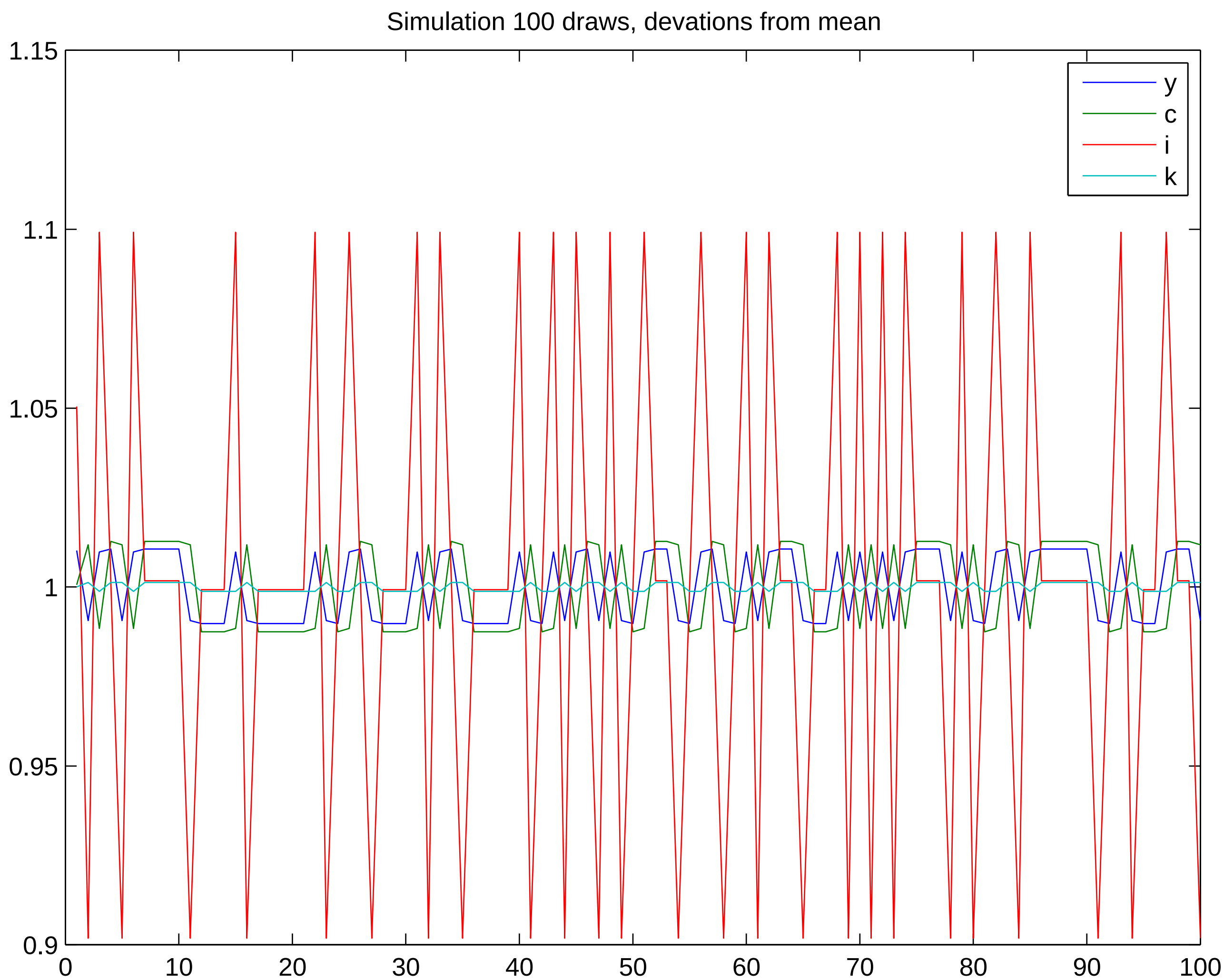
1 Stochastic dynamic programming





30
29.5
29
28.5
28
27.5
250
255
260
265
270
275
280
285
290
295
300
c_t
k_t
c_t = d(k_t,high z_t)
c_t = d(k_t,low z_t)


Simulation 100 draws, devations from mean
1.15
1.1
1.05
1
0.95
0.9
0
10
20
30
40
50
60
70
80
90
100
y
c
i
k

2 dp3.m

```
clear all;
close all;

% Define the structural parameters of the model,
% i.e. policy invariant preference and technology parameters
% alpha : capital's share of output
% beta  : time discount factor
% delta : depreciation rate
% sigma : risk-aversion parameter, also intertemp. subst. param.
alpha = .35;
beta  = .98;
delta = .025;
sigma = 2;


% Number of exogenous states
gz = 2;
% Values of z
z = [4.95
     5.05];
% Probabilites for z
prh = .5;
prl = 1-prh;
% Expected value of z
zbar = prh*z(1) + prl*z(2);


% Find the steady-state level of capital as a function of
%   the structural parameters
kstar = ((1/beta - 1 + delta)/(alpha*zbar))^(1/(alpha-1));

% Define the number of discrete values k can take
gk = 101;
k  = linspace(0.90*kstar,1.10*kstar,gk);


% Compute a (gk x gk x gz) dimensional consumption matrix c
%   for all the (gk x gk x gz) values of k_t, k_t+1 and z_t.
for h = 1 : gk
   for i = 1 : gk
       for j = 1 : gz
            c(h,i,j) = z(j)*k(h)^alpha + (1-delta)*k(h) - k(i);
            if c(h,i,j) < 0
                c(h,i,j) = 0;
            end
            % h is the counter for the endogenous state variable k_t
            % i is the counter for the control variable k_t+1
```

```
            % j is the counter for the exogenous state variable z_t
        end
    end
end

% Compute a (gk x gk x gz) dimensional consumption matrix u
%   for all the (gk x gk x gz) values of k_t, k_t+1 and z_t.
for h = 1 : gk
    for i = 1 : gk
        for j = 1 : gz
            if sigma == 1
                u(h,i,j) = log(c(h,i,j))
            else
                u(h,i,j) = (c(h,i,j)^(1-sigma) - 1)/(1-sigma);
            end
            % h is the counter for the endogenous state variable k_t
            % i is the counter for the control variable k_t+1
            % j is the counter for the exogenous state variable z_t
        end
    end
end



% Define the initial matrix v as a matrix of zeros (could be anything)
v = zeros(gk,gz);

% Set parameters for the loop
convcrit = 1E-6;  % chosen convergence criterion
diff = 1;         % arbitrary initial value greater than convcrit
iter = 0;         % iterations counter

while diff > convcrit
    % for each combination of k_t and gamma_t
    %   find the k_t+1 that maximizes the sum of instantenous utility and
    %   discounted continuation utility
    for h = 1 : gk
        for j = 1 : gz
            Tv(h,j) = max( u(h,:,j) + beta*(prh*v(:,1)' + prl*v(:,2)') );
        end
    end
    iter = iter + 1;
    diff = norm(Tv - v);
    v = Tv;
end


% Find the implicit decision rule for k_t+1 as a function of the state
%   variables k_t and z_t
```

```
for h = 1 : gk
    for j = 1 : gz
        % Using the [ ] syntax for max does not only give the value, but
        %   also the element chosen.
        [Tv,gridpoint] = max(u(h,:,j) + beta*(prh*v(:,1)' + prl*v(:,2)'));
        % Find what grid point of the k vector which is the optimal decision
        kgridrule(h,j) = gridpoint;
        % Find what value for k_t+1 which is the optimal decision
        kdecrule(h,j) = k(gridpoint);
    end
end


% Plot it and save it as a jpg-file
figure
plot(k,kdecrule,k,k);
xlabel('k_t')
ylabel('k_{t+1}')
print -djpeg kdecrule.jpg


% Compute the optimal decision rule for c as a function of the state
%    variables
for h = 1 : gk       % counter for k_t
    for j = 1 : gz   % counter for z_t
        cdecrule(h,j) = z(j)*k(h)^alpha + (1-delta)*k(h) - kdecrule(h,j);
    end
end

figure
plot(k,cdecrule)
xlabel('k_t')
ylabel('c_t')
print -djpeg decrulec.jpg

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%




% Simulation

% Aribrary starting point
enostate = round(gk/2);
ksim(1) = k(enostate);

% Draw a sequence of 100 random variables and use the optimal decision rule
for i = 1 : 100
    draw = rand;
```

```
    if draw < prh
        exostate = 1;
    else
        exostate = 2;
    end

    kprimegrid = kgridrule(enostate,exostate);
    kprime = k(kprimegrid);

    zsim(i) = z(exostate);
    ksim(i+1) = kprime;
    ysim(i) = z(exostate)*ksim(i)^alpha;
    isim(i) = ksim(i+1) - (1-delta)*ksim(i);
    csim(i) = ysim(i) - isim(i);
end

figure
plot(1:100,ysim/mean(ysim),1:100,csim/mean(csim),1:100,isim/mean(isim),1:100,ksim(1:100)/mean(ksim(1:100))
legend('y','c','i','k')
title('Simulation 100 draws, devations from mean')
print -djpeg simulation.jpg
```