Recursive methods
Spring 2007
Henriksen

Problem set 1

1. (a) Write a program that uses the golden section method to find the maximum of the function

$$f(x) = \log(x) - x.$$

(b) Write computer programs to implement piecewise linear interpolations, and Chebyshev collocation. For the latter follow the algorithm on page 223 of Judd.

Apply the algorithms to the function $g(k) = \alpha\beta Ak^{\alpha}$ on $[0, 10]$. Let $\alpha = .3, \beta = .98$ and $A = 1$.

Find the maximum error (in absolute value) for each interpolation scheme on this interval. (To find the maximum error, you can use the program that you wrote in (a).)

2. (a) Consider a model economy where the social planner chooses an infinite sequence of consumption and next period's capital stock $\{c_t, k_{t+1}\}_{t=0}^{\infty}$ in order to

$$\max_{\{c_t, k_{t+1}\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t u(c_t)$$

subject to

$$\begin{aligned} c_t + k_{t+1} &\leq y_t + (1-\delta)k_t, && \forall t \\ c_t, k_t &\geq 0, && \forall t \\ k_0 &> 0. && \text{given} \end{aligned}$$

Assume the following functional forms

$$\begin{aligned} u(c_t) &= \frac{c_t^{1-\sigma}}{1-\sigma}, && \forall t, \sigma > 0 \\ y_t &= F(k_t, 1) = \gamma k_t^{\alpha}, && \forall t, \alpha \in (0,1) \end{aligned}$$

Reformulate this problem as a dynamic programming problem, i.e. write up the Bellman equation.

(b) Write a program where you find the value function defined over a continuous state space using value function iteration with piecewise linear interpolation.

For the interpolation step and the optimization step you might use your algorithms from 1.

(c) Write a program where you find the value function defined over a continuous state space using value function iteration with Chebychev interpolation.

For the interpolation step and the optimization step you might use your algorithms from 1.

See Appendix A for a detailed step-by-step description of the algorithm.

3. Consider the following simple version of the stochastic neoclassical growth model. The economy is populated by a large number of identical agents with total measure one. Each agents has a time endowment of 1 per period. Agents rank life time consumption and leisure according to

$$E\sum_{t=0}^{\infty}\beta^t u\left(c_t, h_t\right),\ \beta\in(0,1)$$

There is a large number of identical firms with aggregate technology given by

$$y_t = e^{z_t}F\left(k_t, h_t\right)$$

where z_t is a discrete state Markov process with transaction probability matrix Π and stationary distribution P. The function F is assumed to be strictly increasing in each of its arguments, strictly concave, constant returns to scale and it also satisfies the INADA conditions.

The technology shock has an unconditional mean of zero and it follows an two-state Markov process; the state space of this Markov chain is $Z=\{-\varepsilon,\varepsilon\}$. Denote the transition matrix Π for this Markov process as

$$\Pi=\begin{pmatrix}\kappa & 1-\kappa\\ 1-\kappa & \kappa\end{pmatrix}.$$

where the (i,j) element $\pi_{i,j}=\Pr\left\{z_{t+1}=z_j \mid z_t=z_i\right\}$.

The law of motion for the aggregate capital stock is given by

$$k_{t+1}=(1-\delta)k_t+i_t$$

where δ is the depreciation rate of capital.

The resource constraint is given by

$$c_t + i_t = y_t.$$

Assume the following functional forms

$$\begin{aligned} u(c_t, h_t) &= \log(c_t) + \psi \log(1 - h_t), \\ F(k_t, h_t) &= k_t^{\alpha} h_t^{1-\alpha} \end{aligned}$$

(a) Write the Bellman's equation for the social planner's problem. State clearly what are the endogenous state variable(s), the exogenous state variable(s) and the control variable(s) in this problem.

(b) Attached please find the Matlab code which solves this problem using the method of discrete value function iteration. Study the code and make sure you understand each step. In particular study the stochastic simulations.

A Value function iteration using Chebychev interpolation

1. Set the order of polynomials used for approximation n.
2. Set the number of collocation points used m. It should be the case that $m \geq n+1$.
   - If $m = n+1$ it is called Chebyshev collocation method, and
   - if $m > n+1$, the method is called Chebyshev regression.
3. Set a convergence criterion ε.
4. Set upperbound and lowerbound of the state space, $[a, b]$.
5. Compute the collocation points $\{x_i\}_{i=1}^m$ implied by Chebyshev collocation points (or extended Chebyshev array) $\{z_i\}_{i=1}^m$
6. We approximate the value function $v(K)$ by:

$$v(K) = \sum_{i=0}^{n} a_i T_i \left(2\frac{x-a}{b-a} - 1 \right)$$

   Notice that the value function is characterized by coefficients $\{a_i\}_{i=0}^n$.
7. Set an initial guess $\left\{a_i^0\right\}_{i=0}^n$. First choose a guess for the level of the value function at $\{x_k\}_{k=1}^m$. Call them $\left\{y_k^0\right\}_{k=1}^m$
8. Compute Chebyshev coefficients $\left\{a_i^0\right\}_{i=0}^n$

$$a_i = \frac{\sum_{k=1}^m y_k^0 T_i(z_k)}{\sum_{k=1}^m T_i(z_k)^2}$$

   where

$$\begin{aligned} T_0(x) &= 1; \qquad T_1(x) = x, \\ T_{i+1}(x) &= 2xT_i(x) - T_{i-1}(x) \qquad i = 1, \ldots, n. \end{aligned}$$

9. Denote the guess of the value function implied by $\left\{a_i^0\right\}_{i=0}^n$ as $v^0(K)$

10. For each $k = 1, 2, \ldots, m$, solve the following problem:

$$\begin{aligned} g_k &= \text{argmax}_{K' \in [a,b]} \left\{ u\left(F\left(K_k, 1\right) + (1-\delta) K_k - K'\right) + \beta v^0\left(K'\right) \right\} \\ &= \text{argmax}_{K' \in [a,b]} \left\{ u\left(F\left(x\left(k\right), 1\right) + (1-\delta) x\left(k\right) - K'\right) + \beta v^0\left(K'\right) \right\} \end{aligned}$$

    It is necessary to use a one-dimension optimization algorithm to find an optimum g_k, e.g. Golden Section Search.

    Once g_k is obtained, use it to update value function. Specifically:

$$y_k^1 = u\left(F\left(K_k, 1\right) + (1-\delta) K_k - g_k\right) + \beta v^0\left(g_k\right)$$

11. Using $\left\{y_k^1\right\}_{i=1}^m$ and Chebyshev collocation method, obtain updated guess for the coefficients $\left\{a_i^1\right\}_{i=0}^n$

12. Compare $\left\{a_i^0\right\}_{i=0}^n$ and $\left\{a_i^1\right\}_{i=0}^n$. In particular, if:

$$\max \left| a_i^0 - a_i^1 \right| < \varepsilon$$

    holds, we are done.

    Treat $\tilde{V}_1(K)$ as the optimal value function.

    Approximate the optimal decision rule given the evaluations at the approximation nodes $\{g_k\}_{k=1}^m$.

13. Otherwise, update the value function by:

$$a_i^0 = a_1^1$$

    and go back to step 9.

B Discrete Stochastic Value Function Iteration

```
% Solving a stochastic neoclassical growth model with elastic labor supply
% by Espen Henriksen


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

alpha   =  .36;
beta    =  .97;
```

```
delta   =  .006;
epsilon =  .022;
kappa   =  .975;
psi     = 1.78;
sigma   = 1;

z  = [-epsilon, epsilon];
Xi = (1/beta - (1 - delta))/alpha;
Pi = [kappa   1-kappa
      1-kappa kappa];

dim = 2;    % Number of values the exogenous state variable might take
hg  = 101;   % Number of grid points for the control variable h
kg  = 201;  % Number of grid points for both the control var and state var k

khratio = ((1/beta - 1 + delta)/alpha)^(1/(alpha-1));
chratio = khratio^alpha - delta*khratio;

hstar = 1/(1 + psi*chratio/((1-alpha)*khratio^alpha));
kstar = khratio*hstar;
cstar = chratio*hstar;

clear khratio chratio;   % if you won't need 'em delete 'em

kgrid = linspace(.85*kstar,1.15*kstar,kg)';
hgrid = linspace(.8*hstar,1.2*hstar,hg)';

c = zeros(hg,kg,dim,kg);
u = zeros(hg,kg,dim,kg);

% i is a counter for the control variable h
% j is a counter for the control variable k'
% m is a counter for the exog. state variable z
% n is a counter for the endogenous state variable k

warning off; % disable warning for taking log of zero (just annoying)
for i = 1 : hg
    for j = 1 : kg
        for m = 1 : dim
            for n = 1 : kg
                c(i,j,m,n) = exp(z(m))*(kgrid(n)^alpha)*hgrid(i)^(1-alpha) + (1-delta)*kgrid(n) - kgrid
                if c(i,j,m,n) < 0
                    c(i,j,m,n) = 0;
                end
                u(i,j,m,n) = log(c(i,j,m,n)) + psi*log(1-hgrid(i));
            end
        end
    end
end
```

```
warning on;   % turn warnings on again
clear c       % free up memory
clear i j m n % clean up


v = zeros(kg,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 m = 1 : dim
        for n = 1 : kg
            objfn(:,:,m,n) = u(:,:,m,n) + beta*(Pi(m,1)*(v(:,1)*ones(1,hg))'+Pi(m,2)*(v(:,2)*ones(1,hg))
            Tv(n,m) = max(max(objfn(:,:,m,n)));
        end
    end
    diff = norm(v-Tv);
    v = Tv;
    iter = iter + 1;
end


for m = 1 : dim
    for n = 1 : kg
        objfn(:,:,m,n) = u(:,:,m,n) + beta*(Pi(m,1)*(v(:,1)*ones(1,hg))'+Pi(m,2)*(v(:,2)*ones(1,hg))');
        [tmp1,x1] = max(objfn(:,:,m,n),[],1);
        [tmp2,x2] = max(tmp1,[],2);
        kgridrule(m,n) = x2;
        hgridrule(m,n) = x1(x2);
        kdecrule(m,n)  = kgrid(kgridrule(m,n));
        hdecrule(m,n)  = hgrid(hgridrule(m,n));
        cdecrule(m,n)  = exp(z(m))*(kgrid(n)^alpha)*hdecrule(m,n)^(1-alpha) + (1-delta)*kgrid(n) - kdec
    end
end

% If you won't need 'em -- delete 'em
clear tmp1 tmp2 x1 x2
clear diff convcrit iter m n
clear objfn Tv
clear u

 figure;plot(kgrid,v);
 title('Value function')
 figure;
 plot(kgrid,kgrid,kgrid,kdecrule);
 title('Decision rules for capital');
 figure;
```

```
 plot(kgrid,cdecrule)
 title('Decision rules for consumption');
 figure;
 plot(kgrid,hdecrule)
 title('Decision rules for labor supply');


% Stochastic simulations

if mod(kg,2) == 0              % modulus after division,in
    endostate = kg/2;
else
    endostate = (kg-1)/2 + 1;
end
exostate = 1;

kprime = kgrid(endostate);

for ctr = 1 : 10000
    kcurr = kprime;
    draw = rand;
    if exostate == 1
        if draw < Pi(1,1)
            exostate = 1;
        else
            exostate = 2;
        end
    else
        if draw < Pi(2,2)
            exostate = 2;
        else
            exostate = 1;
        end
    end

    kprime = kdecrule(exostate,endostate);
    h(ctr,1) = hdecrule(exostate,endostate);

    k(ctr,1) = kcurr;
    i(ctr,1) = kprime - (1-delta)*kcurr;
    y(ctr,1) = exp(z(exostate))*kcurr^alpha*h(ctr,1)^(1-alpha);
    c(ctr,1) = y(ctr,1) - i(ctr,1);
    r(ctr,1) = exp(z(exostate))*alpha*(kcurr/h(ctr,1))^(alpha-1);
    w(ctr,1) = exp(z(exostate))*(1-alpha)*(kcurr/h(ctr,1))^(alpha);

    endostate = kgridrule(exostate,endostate);
end
clear ctr draw
```

```
% Remove the first 1000 draws from the sample
y = y(1000:10000);
i = i(1000:10000);
h = h(1000:10000);
k = k(1000:10000);
c = c(1000:10000);
r = r(1000:10000);
w = w(1000:10000);

% Compute the statistics
ystd = std(y)/mean(y);
istd = std(i)/mean(i);
hstd = std(h)/mean(h);
kstd = std(k)/mean(k);
cstd = std(c)/mean(c);
rstd = std(r)/mean(r);
wstd = std(w)/mean(w);

ystdy = ystd/ystd;
istdy = istd/ystd;
hstdy = hstd/ystd;
kstdy = kstd/ystd;
cstdy = cstd/ystd;
rstdy = rstd/ystd;
wstdy = wstd/ystd;

ycorry = corrcoef(y,y);
icorry = corrcoef(y,i);
hcorry = corrcoef(y,h);
kcorry = corrcoef(y,k);
ccorry = corrcoef(y,c);
rcorry = corrcoef(y,r);
wcorry = corrcoef(w,i);

fprintf(' \n')
fprintf(' \t Mean      \t St.dev.   \t Std rel y \t Cont.corr y \n');
fprintf(' y \t %1.5f   \t %1.5f     \t %1.5f     \t %1.5f       \n', [mean(y) ystd ystdy ycorry(1,2)])
fprintf(' c \t %1.5f   \t %1.5f     \t %1.5f     \t %1.5f       \n', [mean(c) cstd cstdy ccorry(1,2)])
fprintf(' i \t %1.5f   \t %1.5f     \t %1.5f     \t %1.5f       \n', [mean(i) istd istdy icorry(1,2)])
fprintf(' h \t %1.5f   \t %1.5f     \t %1.5f     \t %1.5f       \n', [mean(h) hstd hstdy hcorry(1,2)])
fprintf(' k \t %1.5f   \t %1.5f     \t %1.5f     \t %1.5f       \n', [mean(k) kstd kstdy kcorry(1,2)])
fprintf(' r \t %1.5f   \t %1.5f     \t %1.5f     \t %1.5f       \n', [mean(r) rstd rstdy rcorry(1,2)])
fprintf(' w \t %1.5f   \t %1.5f     \t %1.5f     \t %1.5f       \n', [mean(w) wstd wstdy wcorry(1,2)])
fprintf(' \n')
fprintf(' \n')
fprintf(' \n')
```

```
for ctr = 1 : 9
    tmp = corrcoef(y(ctr:8990+ctr,1),y(5:8995,1));  % because matlab rather stupidly returns the correla
    corrtable(1,ctr) = tmp(2,1);
    tmp = corrcoef(c(ctr:8990+ctr,1),y(5:8995,1));
    corrtable(2,ctr) = tmp(2,1);
    tmp = corrcoef(i(ctr:8990+ctr,1),y(5:8995,1));
    corrtable(3,ctr) = tmp(2,1);
    tmp = corrcoef(h(ctr:8990+ctr,1),y(5:8995,1));
    corrtable(4,ctr) = tmp(2,1);
end
clear tmp ctr

fprintf('       t-4     t-3     t-2     t-1      t      t+1     t+2     t+3     t+4  \n')
fprintf(' y   %1.4f  %1.4f  %1.4f  %1.4f  %1.4f  %1.4f  %1.4f  %1.4f  %1.4f    \n', [corrtable(1,:)])
fprintf(' c   %1.4f  %1.4f  %1.4f  %1.4f  %1.4f  %1.4f  %1.4f  %1.4f  %1.4f    \n', [corrtable(2,:)])
fprintf(' i   %1.4f  %1.4f  %1.4f  %1.4f  %1.4f  %1.4f  %1.4f  %1.4f  %1.4f    \n', [corrtable(3,:)])
fprintf(' h   %1.4f  %1.4f  %1.4f  %1.4f  %1.4f  %1.4f  %1.4f  %1.4f  %1.4f    \n', [corrtable(4,:)])
```