Macroeconomics Final Exam Review

Algorithms

Backward Induction Algorithm

1. discretize the state space “‘{matlab} kgrid(1:n) = linspace(0, kmax, n); xgrid(1:m) = linspace(0, xmax, m); zgrid(1:p) = linspace(0, zmax, p);

````
2. initialize the value function and policy function
```{matlab}
VR(1:n, 1:m, R:T) = 0; % final value is health probability
V(1:n, 1:p, 1:m, 1:R-1) = 0;
coptR(1:n, 1:m, R:T);
koptR(1:n, 1:m, R:T);
copt(1:n, 1:m, 1:p, 1:R-1);
kopt(1:n, 1:m, 1:p, 1:R-1);
lopt(1:n, 1:m, 1:p, 1:R-1);
````

3. starting at the last period, iterate backwards

```
{matlab} for ik = 1:n   for ix = 1:m     koptR(ik, ix, T) = 0;
totres = kgrid(ik)*(1+r)+ss-xgrid(ix);     if totres < cmin
coptR(ik,ix,T) = cmin;       VR(ik, ix, T) = log(cmin);     else
coptR(ik,ix,T) = totres;       VR(ik, ix, T) = log(totres);
end   end end
```

4. define utility function to call during iteration “‘{matlab} function u = ufunc(kp, ik, ix, t) global krid, r, ss, xgrid, beta, phi, VR

cons = krid(ik)*(1+r)+ss-xgrid(ix)-kp; % use find to get kp in grid utiltoday = log(cons); wght = (kp - kgrid(jlo))/(kgrid(jhi) - kgrid(jlo)); Eval = 0; for ixp = 1:m Vint = (1-omega)*VR(jlo, ixp, t+1) + omega*VR(jhi, ixp, t+1); Eval = Eval + phi(ixp, ix)*Vint; end u = utiltoday + beta*Eval; u = -u; end

````
5. iterate backwards for the remaining periods


```{matlab}
for t = T-1:R
  for ik = 1:n
    for ix = 1:m
      totres = kgrid(ik)*(1+r)+ss-xgrid(ix);
      if totres < cmin
        coptR(ik, ix, t) = cmin;
        for ixp = 1:n
          val = val + phi*(ixp, ix)*VR(1, ixp, t+1);
````

```
      end
      VR(ik, ix, t) = log(cmin) + beta*VR(ik, ixp, t+1);
      koptR(ik, ix, t) = 0;
    else if totres > cmin
     [fval, kval] = fminsearch(@kp ufunc(kp,t, ix, ik))
      VR(ik, ix, t) = -fval;
      koptR = kval;
      coptR = totres - optkR;
    end
  end
end
for t = R-1:1
  for iz = 1:v
    for ix = 1:m
      for ik = 1:n
        for il = 1:2 % l = 1, no work, 2, work
          totres = kgrid(ik)*(1+r)-xgrid(ix)+zgrid(iz)*w*(il-1);
          if totres < cmin
            copt(ik, ix, iz, t) = cmin;
            for ixp = 1:n
              val = val + phi*(ixp, ix)*V(1, ixp, t+1);
            end
            V(ik, ix, iz, t) = log(cmin) + beta*V(ik, ixp, t+1);
            kopt(ik, ix, iz, t) = 0;
            lopt(ik, ix, iz, t) = 0;
          else if totres > cmin
           [fval, kval] = fminsearch(@kp ufunc(kp,t, iz, ik))
            VR(ik, ix, iz, t) = -fval;
            kopt = kval;
            copt = totres - optk;
          end
          VT(il) = -fval;
          kt(il) = kval;
        end
      if VT(1) > VT(2)
        V(ik, ix, iz, t) = VT(1);
        kopt(ik, ix, iz, t) = kt(1);
        lopt(ik, ix, iz, t) = 1;
      else if VT(2) > VT(1)
        V(ik, ix, iz, t) = VT(2);
        kopt(ik, ix, iz, t) = kt(2);
        lopt(ik, ix, iz, t) = 2;
      end
    end
  end
end
```

`end`

6. define the global for worker's utility

"'{matlab} function u = ufuncw(kp, t, ik, ix, iz, il); global VR, V, kgrid, r, ss, xgrid, zgrid, beta, phi, theta;

cons = kgrid(ik)+labinc-xgrid(ix)-kp; if il = 1 labinc = 0; disu = 0; else if il = 2 labinc = zgrid(iz)*w; disu = theta; end

utiltoday = log(cons) - disu; % Same process for weight and iterpolation as in retired function

Eval = 0; for ixp = 1:n for izp = 1:v Vint = (1-omega) *V(jlo, ixp, izp, t+1) + omega*V(jhi, ixp, izp, t+1); Eval = Eval + phi(ixp, ix)*Vint; end end

u = utiltoday + beta*Eval; u = -u; "'

7. simulate!

2D Linear Interpolation

1. Fix j^h_{lo}

$$[(1-wk)V(j^k_{lo}, j^h_{lo}) + wkV(j^k_{hi}, j^h_{lo})](1-wh)+$$

2. Fix j^h_{hi}

$$[(1-wk)V(j^k_{lo}, j^h_{hi}) + wkV(j^k_{hi}, j^h_{hi})]wh$$

3. $\exists$ stochastic variable z

$$\mathbb{E}v(k,z) = \sum_{i=1}^{n} p_i V(k, z_i)$$

4. - set n
   - set:

$$\bar{z} = \mu + \lambda \cdot \sigma_z$$

$$\underline{z} = \mu - \lambda \cdot \sigma_z$$

- set $z_2, ..., z_{n-1}$

$$z_1 = \underline{z}, z_2 = z_1 + step,\ z_3 = z_2 + step, ..., z_n = \bar{z}$$

5. $m_i = \frac{z_i + z_{i+1}}{2}$ is the midpoint

$$Pr(z = z_i) = Pr(z \in [m_i, m_{i+1}]) - Pr(m_i < z < m_{i+1})$$

$$= Pr(\frac{z - \mu}{\sigma_z} < \frac{m_{i+1} - \mu}{\sigma_z}) - P(\frac{z - \mu}{\sigma_z} < \frac{m_i - \mu}{\sigma_z})$$

$$= \Phi(\frac{m_{i+1} - \mu}{\sigma_z}) - \Phi(\frac{m_i - \mu}{\sigma_z})$$

$$Pr(z = z_1) = Pr(z < m_1) = \Phi(\frac{m_1 - \mu}{\sigma_z})$$

$$Pr(z = z_n) = Pr(z > m_n) = 1 - \Phi(\frac{m_{n-1} - \mu}{\sigma_z})$$

Tauchen

$$z_{t+1} = \mu(1 - \rho) + \rho z_t + \epsilon_{t+1}$$

$$\epsilon_{t+1} \sim N(0, \sigma_\epsilon^2)$$

Conditional Distribution of z

$$Pr(z_{t+1} = z_i | z_t = z_j) : \mathbb{E}[z_{t+1}|z_t] = \mu(1 - \rho) + \rho z_t var(z_{t+1}|z_t) = \sigma_\epsilon^2$$

Unconditional Distribution of z

$$\mathbb{E}[z] = \mu var(z) = \frac{\sigma_\epsilon^2}{1 - \rho^2}$$

Steps

1. Set n
2. Set $\underline{z} = \mu - \lambda \cdot \sigma_z$ and $\bar{z} = \mu + \lambda \cdot \sigma_z$
3. Set $z_2, ..., z_{n-1} = z_{i=1} + step$ s.t. $step = \frac{2\lambda \cdot \sigma_z}{n-1}$
4. Construct midpoints > Example

$$Pr(z' = z_i | z = z_s) = Pr(m_{i-1} < z' < m_i | z = z_s) = Pr(\frac{z' - (1 - \rho)\mu - \rho z_s}{\sigma_\epsilon} < \frac{m_i - (1 - \rho)\mu - \rho z_s}{\sigma_\epsilon}) - Pr$$

$$= \Phi(\frac{m_i - (1 - \rho)\mu - \rho z_s}{\sigma_\epsilon}) - \Phi(\frac{m_{i-1} - (1 - \rho)\mu - \rho z_s}{\sigma_\epsilon})$$

$$Pr(z' = z_1 | z = z_s) = \Phi(\frac{m_1 - (1 - \rho)\mu - \rho z_s}{\sigma_\epsilon})$$

$$Pr(z' = z_n | z = z_s) = 1 - \Phi(\frac{m_{n-1} - (1 - \rho)\mu - \rho z_s}{\sigma_\epsilon})$$

Tauchen-Husssey

1. Use Gauss-Hermite quadrature to approximate the integral

$$\int_{-\infty}^{\infty} f(z)dz = \sum_{i=1}^{n} \omega_i f(z_i)$$

$$\int\limits_{-\infty}^{\infty} v(z')f(z'|z)dz = \int v(z') \frac{1}{\sqrt{2\pi\sigma_\epsilon^2}} \exp(-\frac{(z' - (1-\rho)\mu - \rho z)^2}{2\sigma_\epsilon^2})dz'$$

$$v(z')f(z'|z) \cdot \frac{f(z'|\alpha)}{f(z'|\alpha)} = \phi(z', z)f(z'|\alpha)$$

Idk what the fuck is going on here lol