```
#include <oxstd.h>
#include <oxdraw.h>
#include <oxfloat.h>
#include <quadpack.h>
#import <maximize>
#include <oxprob.h>


/*****************************************************************************************************/
/* Global variables */
decl alpha=2;
decl lambda=-4;
decl beta=0.99;
decl ibar=8;
decl ps=1;
decl pr=4;
decl c=1;
decl gamma=1/2;
decl mF1,mF0,mS;
/* Simulated choices and states */
decl vPhat; /* Estimted choice-probalities Sx1 */
decl vY; /* Observed choices Nx1 */
decl vSid; /* Observed state IDs Nx1 */
/* State space indices */
enum{iI,iC,iP}; /* Columns identifiers for each state variable */
/* Parameter names */
enum{ilambda}; /* Row identifiers for parameter */

/*****************************************************************************************************/
/* Question 1: Bellman equation */
/*****************************************************************************************************/

/* Choice-specific value-function */
value(amV,const vEV)
{
  decl vU1=alpha*mS[][iC]-mS[][iP];
  decl vU0=alpha*mS[][iC].*(mS[][iI].>0)+lambda*(mS[][iC].>0).*(mS[][iI].==0);
  decl mV=(vU0+beta*mF0*vEV)~(vU1+beta*mF1*vEV);
  amV[0]=mV;
  return 1;
}
/* Expected value function */
emax(aEV,const vEV0)
{
  decl mV;
  value(&mV,vEV0);
  aEV[0]=log(sumr(exp(mV)))+M_EULER;
  return 1;
}
/* CCP mapping */
ccp(aEV,aP,const vP)
{
  decl vU1=alpha*mS[][iC]-mS[][iP];
  decl vU0=alpha*mS[][iC].*(mS[][iI].>0)+lambda*(mS[][iC].>0).*(mS[][iI].==0);
  decl vE1=M_EULER-log(vP);
  decl vE0=M_EULER-log(1-vP);
  decl mF=mF0.*(1-vP)+mF1.*vP;
```

```
  decl vEU=(1-vP).*(vU0+vE0)+vP.*(vU1+vE1);
  decl vEVp=invert(unit(rows(vP))-beta*mF)*vEU;
  decl mV;
  value(&mV,vEVp);
  aP[0]=exp(mV[][1])./sumr(exp(mV));
  aEV[0]=vEVp;
  return 1;
}
/*****************************************************************************************/

/*****************************************************************************************/
/* Question 3: Likelihood function full-solution vs two-step CCP estimator */
/*****************************************************************************************/
lfunc_ccp(const vP, const adFunc, const avScore, const amHessian)
{
  lambda=vP[ilambda];
  decl vCCP,vEV;
  ccp(&vEV,&vCCP,vPhat);
  vCCP=vCCP[vSid];
  decl vL=vCCP.*vY+(1-vCCP).*(1-vY);
  decl LLF;
  LLF=double(sumc(log(vL)));
  adFunc[0]=LLF;
  return 1;
}
lfunc_nfxp(const vP, const adFunc, const avScore, const amHessian)
{
  lambda=vP[ilambda];
  decl vCCP,vCCP0;
  decl it=0;
  decl eps=10^(-10);
  vCCP=vPhat;
  decl vEV=constant(0,rows(mS),1),vEV0;
  /* CCP algorithm */
  do{
    vEV0=vEV;
    vCCP0=vCCP;
    ccp(&vEV,&vCCP,vCCP0);
    it+=1;
  }while(norm(vEV-vEV0)>eps);
  vCCP=vCCP[vSid];
  decl vL=vCCP.*vY+(1-vCCP).*(1-vY);
  decl LLF;
  LLF=double(sumc(log(vL)));
  adFunc[0]=LLF;
  return 1;
}
/*****************************************************************************************/


main()
{
  format(1000);
  decl mPi=<0.75,0.25;0.9,0.1>;
  decl mGamma=(gamma~(1-gamma))|(gamma~(1-gamma));
  println("Gamma: ",mGamma);
  decl vIgrid=range(0,ibar,c)';
```

```
/* Inverted value function */
Question 2: Comparison of true/estimated value function
using frequency estimator */

        I &         C &         P & $P(x)^\ast$ & $\hat{P}
(s)$ \\
        0 &         0 &         4 &     0.534 &     0.525 &
        1 &         0 &         4 &     0.399 &     0.392 &
        2 &         0 &         4 &     0.323 &     0.317 &
        3 &         0 &         4 &     0.272 &      0.28 &
        4 &         0 &         4 &     0.233 &     0.227 &
        5 &         0 &         4 &     0.199 &     0.172 &
        6 &         0 &         4 &     0.163 &     0.176 &
        7 &         0 &         4 &      0.11 &     0.182 &
        8 &         0 &         4 &     0.018 &     0.001 &
        0 &         1 &         4 &     0.881 &     0.885 &
        1 &         1 &         4 &     0.534 &     0.531 &
        2 &         1 &         4 &     0.399 &     0.389 &
        3 &         1 &         4 &     0.323 &     0.327 &
        4 &         1 &         4 &     0.272 &     0.219 &
        5 &         1 &         4 &     0.233 &     0.257 &
        6 &         1 &         4 &     0.199 &     0.164 &
        7 &         1 &         4 &     0.163 &     0.128 &
        8 &         1 &         4 &      0.11 &     0.273 &
        0 &         0 &         1 &      0.96 &     0.999 &
        1 &         0 &         1 &     0.932 &     0.933 &
        2 &         0 &         1 &     0.907 &     0.917 &
        3 &         0 &         1 &     0.884 &      0.92 &
        4 &         0 &         1 &     0.861 &     0.846 &
        5 &         0 &         1 &     0.836 &     0.783 &
        6 &         0 &         1 &     0.801 &     0.667 &
        7 &         0 &         1 &     0.728 &     0.714 &
        8 &         0 &         1 &     0.269 &     0.001 &
        0 &         1 &         1 &     0.993 &     0.999 &
```

```
        1 &         1 &         1 &      0.96 &     0.955 &
        2 &         1 &         1 &     0.932 &     0.948 &
        3 &         1 &         1 &     0.907 &       0.9 &
        4 &         1 &         1 &     0.884 &     0.903 &
        5 &         1 &         1 &     0.861 &     0.824 &
        6 &         1 &         1 &     0.836 &     0.714 &
        7 &         1 &         1 &     0.801 &     0.833 &
        8 &         1 &         1 &     0.728 &     0.667 &

        I &         C &         P &      $EV$ & \hat{EV}$
\\
        0 &         0 &         4 &      61.1 &      60.9
\\
        1 &         0 &         4 &        65 &      64.8
\\
        2 &         0 &         4 &      68.5 &      68.3
\\
        3 &         0 &         4 &      71.7 &      71.4
\\
        4 &         0 &         4 &      74.6 &      74.4
\\
        5 &         0 &         4 &      77.4 &      77.1
\\
        6 &         0 &         4 &        80 &      79.6
\\
        7 &         0 &         4 &      82.3 &      81.7
\\
        8 &         0 &         4 &      84.1 &      83.4
\\
        0 &         1 &         4 &      58.5 &      58.3
\\
        1 &         1 &         4 &      63.1 &      62.9
\\
        2 &         1 &         4 &        67 &      66.8
\\
        3 &         1 &         4 &      70.5 &      70.3
```

```
\\
      4 &       1 &       4 &    73.7 &    73.4
\\
      5 &       1 &       4 &    76.6 &    76.4
\\
      6 &       1 &       4 &    79.4 &    79.1
\\
      7 &       1 &       4 &      82 &    81.6
\\
      8 &       1 &       4 &    84.3 &    83.6
\\
      0 &       0 &       1 &    63.2 &      63
\\
      1 &       0 &       1 &    66.9 &    66.7
\\
      2 &       0 &       1 &    70.2 &      70
\\
      3 &       0 &       1 &    73.3 &      73
\\
      4 &       0 &       1 &    76.1 &    75.8
\\
      5 &       0 &       1 &    78.8 &    78.4
\\
      6 &       0 &       1 &    81.2 &    80.7
\\
      7 &       0 &       1 &    83.3 &    82.7
\\
      8 &       0 &       1 &    84.3 &    83.3
\\
      0 &       1 &       1 &      61 &    60.8
\\
      1 &       1 &       1 &    65.2 &      65
\\
      2 &       1 &       1 &    68.9 &    68.7
\\
      3 &       1 &       1 &    72.2 &      72
```

```
\\
        4 &         1 &         1 &      75.3 &        75
\\
        5 &         1 &         1 &      78.1 &      77.8
\\
        6 &         1 &         1 &      80.8 &      80.4
\\
        7 &         1 &         1 &      83.2 &      82.8
\\
        8 &         1 &         1 &      85.3 &      84.7
\\


//////////////////////////////////////////////////
Question 4: Estimation results

                      true       2-step     True
CCP         NFXP
lambda
-4.0000      -4.0337      -4.0235      -4.0244
Cannot show draw window!
```