```
   Help
#include "hes1d_std.h"
#include "pnl/pnl_vector.h"
#include "pnl/pnl_complex.h"
#include "pnl/pnl_fft.h"
#include "pnl/pnl_mathtools.h"
#include "pnl/pnl_matrix.h"
#include "pnl/pnl_specfun.h"
#include "pnl/pnl_root.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2012+2) //The "#else" part of the code will be freely av
    ailable after the (year of creation of this file + 2)

static int CHK_OPT(AP_CosineBermudan)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(AP_CosineBermudan)(void *Opt, void *Mod, Pricing
    Method *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else

static PnlMatComplex *be;
static int Nnumber, gflag, gM, jdex;
static double glambda, gvbar, geta, gv0,gT, gr, gstrike,
    grho,ga, gb;


void static gauss_legendre_quadrature_weight(double av,
    double bv, PnlVect *x, PnlVect *w)
{
   PnlVect  *x0,*x1, *w1;
    int size=64, i;
         x0= pnl_vect_create_from_zero (128);
    x1=pnl_vect_create_from_list(size,   0.01222369896061
    57641980521,0.03666379096873349333021 53,0.0610819696041395
    681037870,0.08546364050451549863649 80,0.1097942311276437466
    6729747,0.13405919946118778511757 53,0.1582440427142249339
```

```
74755,0.18233430598533718241038 26,0.20631559090207921715405
80,0.23017356422665998641098 66,0.25389396642269432085561 8
0,0.27746262017790440280623 16,0.30086543887767720266715 41,
0.32408843502441337518325 23,0.34711772859763550842616 28,0.
36993955534985902661659 17,0.39254027503326744273564 82,0.41
49063795522750154922739,0.43702450103710416293704 29,0.4588
814198335521954490891,0.48046407240417202585827 57,0.501759
559136144464289606 3,0.52275515205117547845394 79,0.54343830
24128103634441936,0.56379664822661808391443 08,0.5838180216
287630895500389,0.60349045615854862420357 32,0.622802193910
5849107615396,0.64174169256230755715352 49,0.66029763227264
60521059468,0.67845892244771925936775 57,0.6962147083695143
323850866,0.71355437768358741334385 99,0.730467566741908806
4717369,0.74694416679706198116988 24,0.76297433004409472277
97691,0.77854847550641196685049 41,0.79365729476219329024 33
329,0.80829175750791366011964 22,0.82244311695564384246459 4
2,0.83610291506090684711687 53,0.84926298757796896916360 01,
0.86191546893954846059063 23,0.87405279695803179869541 80,0.
88566771734539721740829 24,0.89675328804915818438644 74,0.90
730288340175681392148 59,0.91731019808096053703648 36,0.9267
692508789478433346245,0.93567438827791637578312 68,0.944020
2878302201821211114,0.95180196134126438621779 63,0.95901475
78536999280989185,0.96565436643196526864582 90,0.9717168187
471365809043384,0.97719849146390738716537 44,0.982096108435
7185360247656,0.98640674272458620887123 55,0.99012781849173
43833379303,0.99325711290021293530343 72,0.9957927585349811
868641612,0.99773324862551401988215 74,0.999077459977375895
0119878,0.99982488794713191447360 81);
     w1=pnl_vect_create_from_list(size,0.0244461801962
625182113259,0.02443156909785004505485 86,0.024402355633849
5820932980,0.02435855726469062585326 85,0.02430020016797186
53234426,0.02422731922281524812009 33,0.02413995798901928499
77167,0.02403816868102405263758 73,0.023922012136703455672
4504,0.02379155778100340063878 07,0.023646883584447615143650
14,0.02348807601653591315302 53,0.023315229994062760122415 7
,0.02312844882438702787929 79,0.02292784414368684692041 10,0
.02271353585023646130971 26,0.02248565203274496687182 46,0.0
22244328893799765104629 1,0.02198971066846049143412 21,0.021
721949538052075375261 0,0.02144120553920846013711 19,0.02114
76464682213485370195,0.02084144778075114911358 39,0.0205227
924869600694322850,0.02019187104213004118067 32,0.019848881
```

```
    2328308622199444,0.019494028058706602823021 9,0.01912752360
    99509454865185,0.018749586940544708650919 5,0.0183604439373
    313432212893,0.017960327185008685940196 9,0.017549475827117
    7046487069,0.017128135423111376830681 0,0.01669655780158920
    45890915,0.016255000909785187051657 5,0.0158037286593993468
    589656,0.015343010768865144085990 9,0.014873122602147314252
    3855,0.014394345004166846176823 9,0.013906964132951985244280
    ,0.013411271288616332314489 0,0.012907562739267347220442 8
    ,0.012396139543950922968821 7,0.011877307372740279575891 1,0
    .011351376324080416693281 7,0.010818660739503076247659 6,0.0
    102794790158321571332153,0.009734153415006805863548 3,0.009
    1830098716608743344787,0.008626377798616749704978 8,0.00806
    45898904860579729286,0.007497981925634728687672 0,0.0069268
    925668988135634267,0.006351663161707188787214 3,0.005772637
    5428656985893346,0.005190161832676330205070 8,0.00460458425
    67029551182905,0.004016254983738642313194 3,0.0034255260409
    102157743378,0.002832751471457991095285 7,0.002238288430962
    6187436221,0.001642503018669029538790 9,0.00104581267934034
    87793129,0.000449380960292090376394 3);

        for (i=0; i<size; i++)
        {
            pnl_vect_set (x0, size-i-1, -GET (x1, i));
            pnl_vect_set (w, size-i-1, (bv-av)/2.*GET (w1,
    i));
            pnl_vect_set (x0, size+i, GET (x1, i));
            pnl_vect_set (w, size+i, (bv-av)/2.*GET (w1, i)
    );
        }
        for (i=0; i<2*size; i++)
        {
            pnl_vect_set (x, i, av+(bv-av)/2.*(GET (x0, i)+
    1.));
        }
    pnl_vect_free(&x1);
    pnl_vect_free(&w1);
          pnl_vect_free(&x0);
}
void static fq(double lambda, double vbar, double eta,
    double *result)
{
```

```
  *result= 2.*lambda*vbar/pow(eta,2.)-1.;
}
void static fzeta(double lambda, double eta, double delta,
    double *result)
{
  *result=2.*lambda/((1.0-exp(-lambda*delta))*pow(eta,2.0)
    );
}
void static fdlnv(double sigmat, double sigmas, double t,
    double s, double lambda, double zeta, double q, double *result)
{
  *result= zeta*exp(-zeta*(exp(sigmas)*exp(-lambda*(t-s))+
    exp(sigmat)))*pow(exp(sigmat)/(exp(sigmas)*exp(-lambda*(t-s)
    )) , q/2.)*exp(sigmat)*pnl_bessel_i(q, 2.*zeta*exp(-lambd
    a*(t-s)/2.)*sqrt(exp(sigmas)*exp(sigmat)));
}
static double fdensity(double x, void *p)
{
        double q, zeta, density;
        fq(glambda, gvbar, geta, &q);
        fzeta(glambda, geta, gT, &zeta);
        fdlnv(x, log(gv0), gT, 0.0, glambda, zeta, q, &dens
    ity);
        return density-1.0e-7;
}
void static range_lnv(double lambda, double vbar, double et
    a, double v0, double T, double *av, double *bv)
{
  double q, zeta, vlogmean, x0, x1, x2, roots,epsabs, epsr
    el;
   PnlFunc func;
   int N_max;
    double leftb;
        fq(lambda, vbar, eta, &q);
        fzeta(lambda, eta, T, &zeta);
    vlogmean=log(v0*exp(-lambda*T)+vbar*(1.-exp(-lambda*T)
    ));
        x0 = vlogmean;
    x1 = vlogmean-20./(1.+q);
        x2 = vlogmean+10./(1.+q);
```

```
        func.function = fdensity;
        func.params = NULL;
        epsabs = 1.0e-8;
        epsrel = 1.0e-9;
        N_max=50;

        pnl_root_bisection(&func,x1, x0, epsrel, epsabs, N_
    max, &roots);
        *av = roots;
        leftb=roots;
        pnl_root_bisection(&func, leftb+0.001, x2, epsrel,
    epsabs, N_max, &roots);
        *bv=roots;
}
void static range_x(double r, double lambda, double vbar,
    double eta, double v0, double T, double rho, double S0, double
    K, double *ax, double *bx)
{
  double L=12;
  double c1=r*T+(1.-exp(-lambda*T))*(vbar-v0)/(2.0*lambda)-
    0.5*vbar*T;
  double c2=(1.0/(8.0*pow(lambda,3)))*(eta*lambda*T*exp(-
    lambda*T)*(v0-vbar)*(8.0*lambda*rho-4.0*eta)+lambda*rho*eta*
    (1.0-exp(-lambda*T))*(16.0*vbar-8.0*v0)+2.0*vbar*lambda*T*
    (-4.0*lambda*rho*eta+pow(eta,2)+4*pow(lambda,2))+pow(eta,2
    )*((vbar-2.0*v0)*exp(-2*lambda*T)+vbar*(6.0*exp(-lambda*T)
    -7)+2*v0)+8*pow(lambda,2)*(v0-vbar)*(1-exp(-lambda*T)));
  /*Truncation range*/
  *ax=c1-L*pow(fabs(c2),0.5)+log(S0/K);
  *bx=c1+L*pow(fabs(c2),0.5)+log(S0/K);
 }
void static funcphi( double a, double b, int number, int jp
    oint, double T, int M, double lambda, double rho, double et
    a, double vbar, double miu, PnlVect *x, PnlHmatComplex *tt
    phi)
{
        double q=0.0, zeta=0.0, sigmat, sigmas, omega, delt
    a=T/M;
        int n, i, j;
        int inde[3]={0,0,0};
        dcomplex nv=CZERO,gamma=CZERO, egamma=CZERO;
```

```
    PnlMatComplex *cbess=pnl_mat_complex_create(jpoint,
 jpoint);
    PnlMat *bess=pnl_mat_create(jpoint, jpoint);


    inde[0]=0;
    fq(lambda, vbar, eta, &q);
    fzeta(lambda, eta, delta, &zeta);
    for (j=0,inde[1]=0; j<jpoint; j++,inde[1]++)
    {
          for(i=0,inde[2]=0; i< j; i++,inde[2]++)
          {
              sigmat=GET(x,i);
              sigmas=GET(x,j);
              pnl_hmat_complex_set(ttphi, inde, CRmul(
CONE,zeta*exp(-zeta*(exp(sigmas)*exp(-lambda*T/M)+exp(sigmat)
))*pow(exp(sigmat)/(exp(sigmas)*exp(-lambda*T/M)) , q/2.)*
exp(sigmat)*pnl_mat_get(bess, j, i)));
          }
          for (i=j,inde[2]=j; i< jpoint; i++,inde[2]++)
          {
                 sigmat=GET(x,i);
                 sigmas=GET(x,j);
                 pnl_mat_set(bess, i, j, pnl_bessel_i(
q, 2.*zeta*exp(-lambda*T/M/2.)*sqrt(exp(sigmas)*exp(sigmat)
)));
                 pnl_hmat_complex_set(ttphi, inde, CRmu
l(CONE,zeta*exp(-zeta*(exp(sigmas)*exp(-lambda*T/M)+exp(si
gmat)))*pow(exp(sigmat)/(exp(sigmas)*exp(-lambda*T/M)) , q/2
.)*exp(sigmat)*pnl_mat_get(bess, i, j)));
          }

    }
           for(n=1,inde[0]=1; n<number; n++,inde[0]++)
            {
                omega = n*M_PI/(b-a);
                nv = RCadd(omega*(lambda*rho/eta-1./2.)
, CRmul(CI, 0.5*omega*omega*(1.-rho*rho)));
                gamma = Csqrt( RCsub( pow(lambda,2.),
Cmul(RCmul(2.*pow(eta,2.),nv), CI) ) );
                egamma= Cexp(CRmul(gamma, -T/M));
```

```
                for (j=0,inde[1]=0; j<jpoint; j++,inde[
1]++)
                {
                    for(i=0,inde[2]=0; i< j; i++,inde[2
]++)
                    {
                      sigmat=GET(x,i);
                      sigmas=GET(x,j);
                      pnl_hmat_complex_set(ttphi, inde,
 RCmul( zeta*exp(-zeta*(exp(sigmas)*exp(-lambda*T/M)+exp(
sigmat)))*pow(exp(sigmat)/(exp(sigmas)*exp(-lambda*T/M)) ,
q/2.)*exp(sigmat) , Cmul(Cexp(CRmul(CI, omega*(miu*T/M+rho/
eta*(exp(sigmat)-exp(sigmas)-lambda*vbar*T/M)))), Cmul(Cmul
( pnl_mat_complex_get(cbess,j,i),  Cdiv(CRmul (Cmul(gamma,
 Csqrt(egamma)), exp(lambda*T/M/2.)*(1.-exp(-lambda*T/M)))
 , RCmul(lambda, RCsub(1., egamma))) ), Cexp (RCmul( (exp(
sigmat)+exp(sigmas))/pow(eta,2.), RCsub (lambda*(1.+exp(-
lambda*T/M))/(1.-exp(-lambda*T/M)) , Cdiv( Cmul(gamma, RCad
d(1., egamma)) , RCsub(1., egamma)) ) ) ) )) ));

                    }
                    for (i=0,inde[2]=0; i< jpoint; i++,
inde[2]++)
                    {
                          sigmat=GET(x,i);
                          sigmas=GET(x,j);
                          pnl_mat_complex_set(cbess, i,
 j, pnl_complex_bessel_i (q, RCmul( sqrt(exp(sigmat)*exp(
sigmas))*4., Cdiv( Cmul(gamma, Csqrt(egamma) ), RCmul ( et
a*eta, RCsub (1. , egamma)) ) ) ));

                          pnl_hmat_complex_set(ttphi,
inde, RCmul( zeta*exp(-zeta*(exp(sigmas)*exp(-lambda*T/M)+
exp(sigmat)))*pow(exp(sigmat)/(exp(sigmas)*exp(-lambda*T/M))
 , q/2.)*exp(sigmat) , Cmul(Cexp(CRmul(CI, omega*(miu*T/M+
rho/eta*(exp(sigmat)-exp(sigmas)-lambda*vbar*T/M)))), Cmul(
Cmul( pnl_mat_complex_get(cbess, i, j),  Cdiv(CRmul (Cmul(
gamma, Csqrt(egamma)), exp(lambda*T/M/2.)*(1.-exp(-lambda*T/
M))) , RCmul(lambda, RCsub(1., egamma))) ), Cexp (RCmul( (
exp(sigmat)+exp(sigmas))/pow(eta,2.), RCsub (lambda*(1.+exp(
-lambda*T/M))/(1.-exp(-lambda*T/M)) , Cdiv( Cmul(gamma, RC
```

```
    add(1., egamma)) , RCsub(1., egamma)) ) ) ) )) ));
                        }
                    }
                }
        pnl_mat_complex_free(&cbess);
        pnl_mat_free(&bess);
}
void static fxi(double a, double b, double l, double u,
    int n, double *result)
{
        *result=(n==0)? (exp(u)-exp(l)):1./(1.+pow(n*M_PI/(
    b-a) ,2.))*( (cos(n*M_PI*(u-a)/(b-a)) + n*M_PI/(b-a) * sin(
    n*M_PI*(u-a)/(b-a)))*exp(u) - (cos(n*M_PI*(l-a)/(b-a)) + n*
    M_PI/(b-a) * sin(n*M_PI*(l-a)/(b-a)))*exp(l) );
}
void static fpsi(double a, double b, double l, double u,
    int n, double *result)
{
  *result=(n==0)? (u-l):(sin(n*M_PI*(u-a)/(b-a))- sin(n*M_
    PI*(l-a)/(b-a)))*(b-a)/(n*M_PI);
}
void static fV(double a, double b, double strike, int n,
    int i, int m, int maturity, int flag, PnlMat *chat, PnlMat *
    star, PnlMat *ve)
{
  double xi, psi, l, u, result=0.;
  if (flag==-1)//put option
  {
    if (m==maturity)
    {
      fxi(a, b, a, 0., n, &xi);
      fpsi(a, b, a, 0., n, &psi);
      result = 2./(b-a)*flag*strike*(xi-psi);
    }
    else
    {
        u =(MGET(star, m+1, i)<=0.)? MGET(star, m+1, i):
    0.;
      l =(a<=0.)? a:0.;
      fxi(a, b, l, u, n, &xi);
      fpsi(a, b, l, u, n, &psi);
```

```
      result = MGET(chat, n, i)+ 2./(b-a)*flag*strike*(x
    i-psi);
    }
  }
        else if (flag==1)// call option
        {
                if (m==maturity)
                {
                        fxi(a, b, 0., b, n, &xi);
                        fpsi(a, b, 0., b, n, &psi);
                        result = 2./(b-a)*flag*strike*(xi-
    psi);
                }
                else
                {
                        u =(b>=0.)? b:0.;
                        l =(MGET(star, m+1, i)>=0.)? MGET(
    star, m+1, i):0.;
                        fxi(a, b, l, u, n, &xi);
                        fpsi(a, b, l, u, n, &psi);
                        result = MGET(chat, n, i) + 2./(b-
    a)*flag*strike*(xi-psi);
                }
        }
        pnl_mat_set(ve, n, i, result);
}
void static funcbeta(int n, int j, int jpoint, PnlVect *w,
    PnlHmatComplex *ttphi, PnlMat *v)
{
        int i;
        int inde[3]={0,0,0};
         dcomplex sum=CZERO;
        inde[0]=n;
        inde[1]=j;

        for(i=0; i<jpoint; i++)
        {
                inde[2]=i;
                sum = Cadd (sum, RCmul(pnl_vect_get(w, i)*
    pnl_mat_get(v, n, i), pnl_hmat_complex_get(ttphi, inde)));
        }
```

```
        pnl_mat_complex_set(be, n, j, sum);
}

void static mbasic(double a, double b, double l, double u,
    PnlVectComplex  *mj)
{
    int n;
    pnl_vect_complex_set(mj, 0, RCmul((u-l)*M_PI/(b-a), CI)
    );
    for (n=1;n<2*Nnumber+1;n++)
    {
        pnl_vect_complex_set(mj, n, CRdiv(Csub(Cexp(RCmul(
    n*(u-a)*M_PI/(b-a) ,CI)) ,Cexp(RCmul(n*(l-a)*M_PI/(b-a),
    CI))) , n));
    }
}

void static c_fft(int j, int number, double T, double M,
    double a, double b, double l, double u, double r, PnlVectCompl
    ex  *mj, PnlMatComplex *beta, PnlMat *chat)
{
    PnlVectComplex *ms=pnl_vect_complex_create(2*number);
    PnlVectComplex *mc=pnl_vect_complex_create(2*number);
    PnlVectComplex *us=pnl_vect_complex_create(2*number);
    PnlVectComplex *ivector1 = pnl_vect_complex_create (2*
    number);
    PnlVectComplex *ivector2 = pnl_vect_complex_create (2*
    number);
    int n;

    for(n=0; n<number;n++)
    {
          pnl_vect_complex_set(ms, n, RCmul(-1, Conj(pnl_
    vect_complex_get(mj, n))));
          pnl_vect_complex_set(us, n, pnl_mat_complex_get(
    beta, n, j));
          pnl_vect_complex_set(mc, n, pnl_vect_complex_get(
    mj, 2*number-1-n));
    }
    pnl_vect_complex_set(us, 0, RCmul(0.5, pnl_mat_
    complex_get(beta, 0, j)));
```

```
    for (n=Nnumber; n<2*number; n++)
    {
          pnl_vect_complex_set(ms, n, pnl_vect_complex_get(
    mj, 2*number-n));
          pnl_vect_complex_set(us, n, CZERO);
          pnl_vect_complex_set(mc, n, pnl_vect_complex_get(
    mj, 2*number-1-n));
    }
    pnl_vect_complex_set(ms, number,CZERO);

      pnl_fft_inplace (us);
      pnl_fft_inplace (ms);
      pnl_fft_inplace (mc);

      for (n=0;n<2*number;n++)
      {
          pnl_vect_complex_set(ivector1, n,Cmul(pnl_vect_
    complex_get(ms, n),pnl_vect_complex_get(us,n)));
          pnl_vect_complex_set(ivector2, n,Cmul(pnl_vect_
    complex_get(mc, n),pnl_vect_complex_get(us,n)));
      }
      for (n=0;n<number;n++)
      {
          pnl_vect_complex_set(ivector2, 2*n+1, RCmul(-1.,
    pnl_vect_complex_get(ivector2, 2*n+1)));
      }
      pnl_ifft_inplace (ivector1);
      pnl_ifft_inplace (ivector2);
      for(n=0; n<number; n++)
      {
            pnl_mat_set(chat, n, j, exp(-r*T/M)/M_PI*(Cimag
    (pnl_vect_complex_get(ivector1, n))+Cimag(pnl_vect_
    complex_get(ivector2, Nnumber-1-n)))) ;
      }

    pnl_vect_complex_free(&ms);
    pnl_vect_complex_free(&mc);
    pnl_vect_complex_free(&us);
    pnl_vect_complex_free(&ivector1);
    pnl_vect_complex_free(&ivector2);
}
```

```
static void fdf(double x, double *f, double *df, void *p)
{
  int n;
        dcomplex  sum=CZERO, sumd=CZERO;
        double g=0.0, dg=0.0;
        sum= RCmul(0.5, pnl_mat_complex_get(be, 0, jdex));
        sumd = CZERO;
        for(n=1; n<Nnumber; n++)
        {
                sum= Cadd(sum, Cmul(pnl_mat_complex_get(be,
     n, jdex), Cexp(CRmul(CI,n*M_PI*(x-ga)/(gb-ga) ))));
                sumd= Cadd(sumd, Cmul(pnl_mat_complex_get(
    be, n, jdex), Cmul( Cexp(CRmul( CI,n*M_PI*(x-ga)/(gb-ga) )),
     CRmul(CI ,n*M_PI/(gb-ga)))));
        }
        g=(gflag*(exp(x)-1.0)>=0.0 )? gflag*(exp(x)-1.0): 0
    .0;
        *f = exp(-gr*gT/gM)*Creal(sum)-gstrike*g ;
        dg=(gflag*(exp(x)-1.0)>=0.0 )? gflag*exp(x): 0.0;
        *df = exp(-gr*gT/gM)*Creal(sumd)-gstrike*dg;
        if(fabs(exp(-gr*gT/gM)*Creal(sumd)-gstrike*dg)<1E-9
    )
        {
          printf("derivative is zero!!   sumd=%f+%f, dg=%f,
     %f-%f=%f, df=%f{n",sumd.r, sumd.i, dg, exp(-gr*gT/gM)*Cr
    eal(sum),exp(-gr*gT/gM)*Creal(sum)-gstrike*g , exp(-gr*gT/
    gM)*Creal(sumd), gstrike*dg);
        }
}
void static newton_root_find(int m, int j, PnlMat *star)
{
        double x0=log(1.), r;
        PnlFuncDFunc func;
        static double epsabs = 0.0000001;
        static double epsrel = 0.00000001;
        static int N_max = 10;

        if (m==gM)       x0 = log(1.);
        else    x0=MGET(star, m+1, j);
        func.function = fdf;
```

```
        func.params = NULL;

         pnl_root_newton(&func, x0, epsrel, epsabs, N_max,
    &r);
         MLET(star, m, j) = r;
}

/*double fdf(double x, void *p)
{
        int n;
        dcomplex  sum=CZERO;
        double g=0.0;
        sum= RCmul(0.5, pnl_mat_complex_get(be, 0, jdex));
        for(n=1; n<Nnumber; n++)
        {
                sum= Cadd(sum, Cmul(pnl_mat_complex_get(be,
     n, jdex), Cexp(CRmul(CI,n*M_PI*(x-ga)/(gb-ga) ))));
        }
        g=(gflag*(exp(x)-1.0)>=0.0 )? gflag*(exp(x)-1.0): 0
    .0;
//          printf("x=%f, c=%f, g=%f, c-g=%f {n", x*100.,
    exp(-gr*gT/gM)*Creal(sum), gstrike*g, exp(-gr*gT/gM)*Creal(
    sum)-gstrike*g);
        return exp(-gr*gT/gM)*Creal(sum)-gstrike*g;
}
void static bisection_root_find(int m, int j)
{
  double x1, x2, r=0.0;
  PnlFunc func;
  int status;

  x1 = ga;
  x2 = gb;
  func.function = fdf;
  func.params = NULL;
  static double epsabs = 0.000001;
  static double epsrel = 0.000001;
  static int N_max = 1000;
  printf("x1=%f,x2=%f   ",x1, x2);

  if (m==gM)  status = pnl_root_bisection(&func, x1, 0.0000
```

```
    1, epsrel, epsabs, N_max, &r);
  else status = pnl_root_bisection(&func, x1,  MGET(star,
    m+1, j), epsrel, epsabs, N_max, &r);
   printf("m=%d, j=%d, bisection : root is %f, status==OK %
    d{n", m, j, r, status==OK);
  MLET(star, m, j) = r;
}
*/
void static Bermuda_Heston(double S0, double strike,
    double T, double r, double divident, double lambda, double eta,
    double vbar, double v0, double rho, int M, int flag, int number,
     int jpoint, double *price)
{
          dcomplex suml=CZERO, sumr=CZERO, sum=CZERO;
          int m, i, j, n;
          double av, bv, ax, bx, c3=0.0, cl, cr, lnS0, miu;
    int dims[3]={number,jpoint,jpoint};
    int index[3]={0,0,0};
          PnlHmatComplex* tphi = pnl_hmat_complex_create(3,
     dims );
          PnlVect *x= pnl_vect_create_from_zero (jpoint);
          PnlVect *w= pnl_vect_create_from_zero (jpoint);

          PnlVectComplex *mj= pnl_vect_complex_create(2*
    number+1);
          PnlMat *star= pnl_mat_create(M+1,jpoint);
          PnlMat *chat = pnl_mat_create(number, jpoint);

          PnlMat *ve = pnl_mat_create(number, jpoint);

          be = pnl_mat_complex_create(number, jpoint);

          range_lnv(lambda, vbar, eta, v0, T, &av, &bv);
          range_x(r, lambda, vbar, eta, v0, T, rho, S0,
    strike, &ax, &bx);

          gauss_legendre_quadrature_weight(av,bv, x, w);

          lnS0=log(S0/strike);
          miu=r-divident;
          ga=ax;
```

```
      gb=bx;
           for(n=0; n<number; n++)
            {
  fV(ga, gb,strike, n, 0, M, M, flag, chat, star,
ve);
                   for (i=1; i< jpoint; i++)
                   {
                      pnl_mat_set(ve, n, i, pnl_mat_
get(ve, n, 0));
                   }
            }
           funcphi(ax, bx, number, jpoint, T, M, lambd
a, rho, eta, vbar, miu, x, tphi);
           for(n=0,index[0]=0; n<number; n++,index[0]++
)
            {
                for (j=0,index[1]=0; j<jpoint; j++,ind
ex[1]++)
                {
                   funcbeta(n, j, jpoint, w, tphi, ve);

                }
            }

            for(j=0; j<jpoint; j++)
            {
                 jdex=j;
                 newton_root_find(M, j, star);// bisect
ion_root_find(m, j);//
                 mbasic(ax, bx, MGET(star,M, j), bx, mj
);
                 c_fft(j, number, T, M, ax, bx, MGET(
star,M, j) , bx, r, mj, be, chat);
            }
      for (m=M-1; m>1; m--)
      {
            for(n=0; n<number; n++)
            {
                for (i=0; i<jpoint; i++)
                {
                       fV(ax, bx,strike, n, i, m, M,
```

```
flag, chat, star, ve);
                }
            }
            for(n=0,index[0]=0; n<number; n++,index[0]+
+)
            {
                for (j=0,index[1]=0; j<jpoint; j++,ind
ex[1]++)
                {
                    funcbeta(n, j, jpoint, w, tphi, ve)
;
                }
            }
            for(j=0; j<jpoint; j++)
            {
                 jdex=j;
                 newton_root_find(m, j, star);// bisect
ion_root_find(m, j);//
                 mbasic(ax, bx, MGET(star,m, j), bx, mj
);
                 c_fft(j, number, T, M, ax, bx, MGET(
star,m, j) , bx, r, mj, be, chat);
            }
       }

       if (log(v0)>=pnl_vect_get(x,jpoint-1) )
       {
              for (i=0; i< jpoint; i++)
               {
                      fV(ax, bx,strike, 0, i, 1, M,
flag, chat, star, ve);
               }
              funcbeta(0, jpoint-1, jpoint, w, tphi,
ve);
              sum = RCmul(0.5, Cmul( pnl_mat_complex_g
et(be, 0, jpoint-1), CONE));
              for (n=1; n< number; n++)
              {
                  for (i=0; i< jpoint; i++)
                  {
                      fV(ax, bx,strike, n, i, 1, M,
```

```
flag, chat, star, ve);
                    }
                   funcbeta(n, jpoint-1, jpoint, w, tp
hi, ve);
                     sum= Cadd(sum, Cmul(pnl_mat_
complex_get(be, n, jpoint-1), Cexp(CRmul(CI,n*M_PI*(lnS0-ax)/(bx-
ax) ))));
                }
                c3 = exp(-r*T/M)*Creal(sum);
        }
        else
        {
                 for (j=0; j< jpoint-1; j++)
                 {
                       if (log(v0)==pnl_vect_get(x,j))
                       {
                              for (i=0; i< jpoint; i++)
                              {
                                     fV(ax, bx,strike, 0
, i, 1, M, flag, chat, star, ve);
                               }
                               funcbeta(0, j, jpoint, w,
 tphi, ve);
                              sum = RCmul(0.5, Cmul(  pn
l_mat_complex_get(be, 0, j), CONE));

                             for (n=1; n< number; n++)
                             {
                                   for (i=0; i< jpoint;
i++)
                                   {
                                       fV(ax, bx,strike,
 n, i, 1, M, flag, chat, star, ve);
                                   }
                                   funcbeta(n, j, jpoint
, w, tphi, ve);
                                   sum= Cadd(sum, Cmul(
pnl_mat_complex_get(be, n, j), Cexp(CRmul(CI,n*M_PI*(lnS0-
ax)/(bx-ax) ))));
                             }
                             c3 = exp(-r*T/M)*Creal(sum)
```

```
;
                        }
                        else if (log(v0)<pnl_vect_get(x,
j+1) && log(v0)>pnl_vect_get(x,j))
                        {
                              for (i=0; i< jpoint; i++)
                               {
                                     fV(ax, bx,strike, 0
, i, 1, M, flag, chat, star, ve);
                                }
                               funcbeta(0, j, jpoint, w,
tphi, ve);
                               funcbeta(0, j+1, jpoint,
w, tphi, ve);
                              suml = RCmul(0.5, Cmul(  pn
l_mat_complex_get(be, 0, j), CONE));
                              sumr = RCmul(0.5, Cmul(  pn
l_mat_complex_get(be, 0, j+1), CONE));

                              for (n=1; n< number; n++)
                              {
                                   for (i=0; i< jpoint;
i++)
                                    {
                                        fV(ax, bx,strike,
 n, i, 1, M, flag, chat, star, ve);
                                    }
                                    funcbeta(n, j, jpoint
, w, tphi, ve);
                                    funcbeta(n, j+1, jpoi
nt, w, tphi, ve);
                                    suml= Cadd(suml, Cmul
(pnl_mat_complex_get(be, n, j)  , Cexp(CRmul(CI,n*M_PI*(ln
S0-ax)/(bx-ax) ))));
                                    sumr= Cadd(sumr, Cmul
(pnl_mat_complex_get(be, n, j+1)  , Cexp(CRmul(CI,n*M_PI*(
lnS0-ax)/(bx-ax) ))));
                              }
                              cl = exp(-r*T/M)*suml.r;
                              cr = exp(-r*T/M)*sumr.r;
                              c3 = cl + (log(v0)-pnl_vec
```

```
    t_get(x,j))/(pnl_vect_get(x,j+1)-pnl_vect_get(x,j))*(cr-cl)
    ;
                          }
                    }
           }
          *price=c3;
  pnl_hmat_complex_free(&tphi);
        pnl_vect_free(&x);
        pnl_vect_free(&w);
        pnl_vect_complex_free(&mj);
        pnl_mat_free(&chat);
        pnl_mat_free(&ve);
        pnl_mat_free(&star);
}


static int Cosine_Bermudan(double S0, double strike,
    double T, double r, double q, double  v0, double vbar, double
    lambda, double eta, double rho, int iscall, int M,double *
    price)
{
  int N, J;

  //Fixed Parameters. Don't modify.
  N=(int)pow(2,7);
  J=(int)pow(2,7);

  gflag=iscall;
  gstrike= strike;
  gT=T;
  gr=r;
  glambda=lambda;
  geta=eta;
  gvbar=vbar;
  gv0=v0;
  grho=rho;
  gM=M;
  Nnumber=N;

  Bermuda_Heston(S0,strike, T, r, q,lambda, eta, vbar, v0,
    rho, M,iscall,N,J,price);
```

```
  pnl_mat_complex_free(&be);

  return OK;
}

int CALC(AP_CosineBermudan)(void *Opt, void *Mod, Pricing
    Method *Met)
{
  TYPEOPT* ptOpt=(TYPEOPT*)Opt;
  TYPEMOD* ptMod=(TYPEMOD*)Mod;
  double r,divid;
  int iscall;

  iscall =-1;//Put Case
  if (ptOpt->PayOff.Val.V_NUMFUNC_1->Compute == &Call) is
    call = 1;

  r=log(1.+ptMod->R.Val.V_DOUBLE/100.);
  divid=log(1.+ptMod->Divid.Val.V_DOUBLE/100.);
  Met->Res[1].Val.V_DOUBLE = 0.;


  return Cosine_Bermudan(ptMod->S0.Val.V_PDOUBLE,
                         ptOpt->PayOff.Val.V_NUMFUNC_1->Par[
    0].Val.V_PDOUBLE,
                         ptOpt->Maturity.Val.V_DATE-ptMod->
    T.Val.V_DATE, r, divid,
                         ptMod->Sigma0.Val.V_PDOUBLE,
                         ptMod->LongRunVariance.Val.V_PDOUB
    LE,
                         ptMod->MeanReversion.hal.V_PDOUBLE,
                         ptMod->Sigma.Val.V_PDOUBLE,
                         ptMod->Rho.Val.V_PDOUBLE,
                         iscall,
                         Met->Par[0].Val.V_INT,
                         &(Met->Res[0].Val.V_DOUBLE));
}

static int CHK_OPT(AP_CosineBermudan)(void *Opt, void *Mod)
{
```

```
  if ((strcmp( ((Option*)Opt)->Name,"CallAmer")==0)||
      (strcmp( ((Option*)Opt)->Name,"PutAmer")==0))
    return OK;

  return  WRONG;
}

#endif

static int MET(Init)(PricingMethod *Met,Option *Opt)
{
  if ( Met->init == 0 )
    {
      Met->Par[0].Val.V_INT=20;
      Met->init = 1;
      Met->HelpFilenameHint = "ap_cosine_bermhes1d";
    }
  return OK;
}


PricingMethod MET(AP_CosineBermudan)=
{
  "AP_CosineBermudan",
  {{"Bermudan Steps",INT,{100},ALLOW},{" ",PREMIA_NULLTYPE,
    {0},FORBID}},
  CALC(AP_CosineBermudan),
  {{"Price",DOUBLE,{100},FORBID},
   {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CHK_OPT(AP_CosineBermudan),
  CHK_ok,
  MET(Init)
};
```

References