Help

```
#include  "
href../../../mod/doublehes1d/doublehes1d_std/doublehes1d_std_h_src.pdfdoublehes1

# include <pnl/pnl_mathtools.h>
# include <pnl/pnl_complex.h>
# include <pnl/pnl_vector.h>
# include <pnl/pnl_matrix.h>
# include <pnl/pnl_specfun.h>
# include <pnl/pnl_cdf.h>

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2020+2) //The "#els
static int CHK_OPT(AP_ZhengFeng_DoubleHeston)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(AP_ZhengFeng_DoubleHeston)(void *Opt, void *Mod, PricingMethod *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else

//define pi M_PI


static void Cal_CoefAB(double V1, double V2, double dividend, double r, double r
{
    *A1  = (V1+V2+2.*(dividend-r))/sqrt(V1+V2);
    *A21 = rho1*sigma1*V1;
    *A22 = rho2*sigma2*V2;
    *A31 = k1*(theta1-V1);
    *A32 = k2*(theta2-V2);
    *A41 = pow(sigma1, 2.)*V1;
    *A42 = pow(sigma2, 2.)*V2;
    *B1  = -2./sqrt(V1+V2);
    *B2  = 1./pow((V1+V2), 3/2.);
    *B4  = -1./(V1+V2)/2.;
    *B6  = 1./4./pow((V1+V2),2.);
    *A1V1= 1./sqrt(V1+V2)-(1/2.)*(V1+V2+2.*dividend-2.*r)/pow((V1+V2),(3/2));
    *A1V1V1 =  -pow(V1 + V2, -3. / 2.) + 3. / 4. * (V1 + V2 + (2. * dividend) -
```

```
    *A1V2= pow(V1 + V2, -1./2.) - (V1 + V2 +  (2. * dividend) -  (2. * r)) * pow
    *A1V2V2 = -pow(V1 + V2, -3. / 2.) + 3. / 4. * (V1 + V2 +  (2. * dividend) -
    *A21V1   =rho1*sigma1;
    *A22V2   = rho2*sigma2;
    *A31V1  = -k1;
    *A32V2  = -k2;
    *A41V1  = pow(sigma1, 2.);
    *A42V2  = pow(sigma2, 2.);
    *B1V1 = pow(V1 + V2, 3. / 2.);
    *B1V1V1 = -1.5 * pow(V1 + V2, -5. / 2.);
    *B2V1  = -3. / 2. * pow(V1 + V2, -5. / 2.);
    *B2V1V1= 15. / 4. * pow(V1 + V2, -7. / 2.);
    *B4V1  = -2. * pow(0.2e1 * V1 + 0.2e1 * V2, -0.2e1);
    *B4V1V1= 8. * pow(0.2e1 * V1 + 0.2e1 * V2, -0.3e1);
    *B6V1  = -0.500e0 * pow(V1 + V2, -0.3e1);
    *B6V1V1=  0.150e1 * pow(V1 + V2, -0.4e1);
}

static void Cal_CoefD(double V1, double V2, double dividend, double r, double rh
{
    *D0  = -(A21*B2*Cn1 + A22*B2*Cn1);
    *D1  = A1*Cn1 + (A21*B1*Cn1V1+ A21*B2*Cn1 + A22*B1*Cn1V2+ A22*B2*Cn1);
    *D2 = A1+ A21*B2+A22*B2;
    *D3 = -(A21*B2+A22*B2);
    *D4  = (A31*2.*Cn2V1 +A32*2.*Cn2V2 + A41*Cn2V1V1+ A42*Cn2V2V2) -2.*r*Cn2;
    *D5 = -(A31*B4*Cn2+ A32*B4*Cn2+ A41*2.*B4*Cn2V1 + A42*2.*B4*Cn2V2 + A41*B6*C
    *D6 = A41*B6*Cn2 +A42*B6*Cn2;
    *D7 = A31*B4*Cn2 + A32*B4*Cn2 + A41*2.*B4*Cn2V1 + 2.*A42*B4*Cn2V2 + A41*3.*B
    *D8 = A31*B4+ A32*B4+ 3.*A41*B6+ 3.*A42*B6;
    *D9 = A41*B6 + A42*B6;

    *D0V1  = -(A21V1*B2*Cn1 + A21*B2V1*Cn1 + A21*B2*Cn1V1 +A22*B2V1*Cn1+A22*B2*C
    *D0V2  = -(A21*B2V1*Cn1 + A21*B2*Cn1V2 +A22V2*B2*Cn1 +A22*B2V1*Cn1 +A22*B2*C
    *D0V1V1=-(A21V1*B2V1*Cn1 +A21V1*B2*Cn1V1 + A21V1*B2V1*Cn1 + A21*B2V1V1*Cn1+
    *D0V2V2=-(A21*B2V1V1*Cn1 + A21*B2V1*Cn1V2 +A21*B2V1*Cn1V2 +A21*B2*Cn1V2V2 +A

}


static void Cal_fg(int N, PnlMat *fni, PnlMat *gni)
{
```

```
    int n, m, i;
    for (n=1; n<N+1; n++)
    {
        m = (n-n%2)/2;
        pnl_mat_set(fni, n, 0, 1.);
        pnl_mat_set(gni, n, 0, 1.);
        for (i=0; i<m+1; i++)
        {
            pnl_mat_set(fni, n, i+1,  (n-2.*i)*(n-2.*i-1.)/(2.*i+2.)*pnl_mat_get
            if ((2.*n-2.*i-2.)<0.0000000001)
            {
                pnl_mat_set(gni, n, i+1, 0.);
            }
            else
            {
                pnl_mat_set(gni, n, i+1, ((n-1.-2.*i)*(n-2.-2.*i)*pnl_mat_get(gn
            }
        }
    }
}


static void Cal_CdfPdf(double y, double *cdf, double *pdf)
{
    double q, mean, sd, bound;
    int which, status;
    which=1;
    mean=0.0;
    sd=1.0;
    pnl_cdf_nor(&which, cdf,&q,&y,&mean,&sd,&status,&bound);
    *pdf = 1./sqrt(2.*M_PI)*exp(-pow(y, 2.)/2.);
}

static void Cal_polynomials(double y, int n, int N, PnlMat *fni, PnlMat *gni, Pn
{
    double result1, result2, result3, result4, result3V1, result4V1, result3V2,
    result1=0.0; result2=0.0; result3=0.0; result4=0.0;
    result3V1=0.0; result4V1=0.0; result3V2=0.0; result4V2=0.0;  result3V1V1=0.0
    int i, m;
    m = (n-n%2)/2;
    for (i= 0; i< m+1; i++)
```

```
{
    if (n==N)
    {
        result1 = result1+ pnl_mat_get(fni, n, i)*pow(y, n-2.*i);
        if (n-2.*i-1>=0) {
            result2 = result2+ pnl_mat_get(gni, n, i)*pow(y, n-2.*i-1);
        }
        if (n-2.*i-2>=0) {
            result3 = result3+ pnl_mat_get(hni, n, i)*pow(y, n-2.*i-2);
        }
        if (n-2.*i-3>=0) {
            result4 = result4+ pnl_mat_get(kni, n, i)*pow(y, n-2.*i-3);
        }
    }
    else
    {
        result1 = result1+ pnl_mat_get(fni, n, i)*pow(y, n-2.*i);
        if (n-2.*i-1>=0) {
            result2 = result2+ pnl_mat_get(gni, n, i)*pow(y, n-2.*i-1);
        }
        if (n-2.*i-2>=0) {
            result3 = result3+ pnl_mat_get(hni, n, i)*pow(y, n-2.*i-2);
            result3V1=result3V1 + pnl_mat_get(hniV1, n, i)*pow(y, n-2.*i-2);
            result3V2=result3V2 + pnl_mat_get(hniV2, n, i)*pow(y, n-2.*i-2);
            result3V1V1=result3V1V1 + pnl_mat_get(hniV1V1, n, i)*pow(y, n-2.
            result3V2V2=result3V2V2 + pnl_mat_get(hniV2V2, n, i)*pow(y, n-2.
            result3V1V2=result3V1V2 + pnl_mat_get(hniV1V2, n, i)*pow(y, n-2.
            result3V2V1=result3V2V1 + pnl_mat_get(hniV2V1, n, i)*pow(y, n-2.
        }
        if (n-2.*i-3>=0) {
            result4 = result4+ pnl_mat_get(kni, n, i)*pow(y, n-2.*i-3);
            result4V1=result4V1 + pnl_mat_get(kniV1, n, i)*pow(y, n-2.*i-3);
            result4V2=result4V2 + pnl_mat_get(kniV2, n, i)*pow(y, n-2.*i-3);
            result4V1V1=result4V1V1 + pnl_mat_get(kniV1V1, n, i)*pow(y, n-2.
            result4V2V2=result4V2V2 + pnl_mat_get(kniV2V2, n, i)*pow(y, n-2.
            result4V1V2=result4V1V2 + pnl_mat_get(kniV1V2, n, i)*pow(y, n-2.
            result4V2V1=result4V2V1 + pnl_mat_get(kniV2V1, n, i)*pow(y, n-2.
        }
    }
}
```

```
    *pn0= result1;
    *qn0= result2;
    *pn1= result3;
    *qn1= result4;
    *pn1V1= result3V1;
    *qn1V1= result4V1;
    *pn1V2= result3V2;
    *qn1V2= result4V2;
    *pn1V2= result3V1V1;
    *qn1V2= result4V1V1;
    *pn1V2= result3V2V2;
    *qn1V2= result4V2V2;
}

static void Cal_C1(int n,  double y, double V1, double V2, double strike, double
{
    *Cn1=strike *pow(V1+V2, 1/2.)*y/(y*cdf+pdf) ;
    *Cn1V1=1/2.*strike *pow(V1+V2, -1/2.)*y/(y*cdf+pdf) ;
    *Cn1V1V1= -1/4.*strike *pow(V1+V2, -3/2.)*y/(y*cdf+pdf);
    *Cn1V1V1V1=1./2.*(-1/2.)*(-3./2.)*strike*pow(V1+V2, -5./2.)*y/(pn0 *cdf + qn
}
static void Cal_C2(int n,  double y, double V1, double V2, double strike, double
{
    *Cn2 = (pow(-1., n+1)/pnl_sf_fact(n)* strike *pow(V1+V2, n/2.)*pow(y,n)-pn1*
    *Cn2V1= (pow(-1., n+1)/pnl_sf_fact(n)* n/2.*strike*pow(V1+V2, n/2.-1.)*pow(y
    *Cn2V2= (pow(-1., n+1)/pnl_sf_fact(n)* n/2.*strike*pow(V1+V2, n/2.-1.)*pow(y
    *Cn2V1V1=( pow(-1., n+1)/pnl_sf_fact(n)* n/2.*(n/2.-1.)*strike*pow(V1+V2, n/
    *Cn2V2V2=( pow(-1., n+1)/pnl_sf_fact(n)* n/2.*(n/2.-1.)*strike*pow(V1+V2, n/
    *Cn2V1V2=( pow(-1., n+1)/pnl_sf_fact(n)* n/2.*(n/2.-1.)*strike*pow(V1+V2, n/
    *Cn2V2V1=( pow(-1., n+1)/pnl_sf_fact(n)* n/2.*(n/2.-1.)*strike*pow(V1+V2, n/
}
void Cal_C3(int n,  double y, double V1, double V2, double strike, double cdf, d
{
    *Cn3 = (pow(-1., n+1)/pnl_sf_fact(n)* strike *pow(V1+V2, n/2.)*pow(y,n)-pn1*
    *Cn3V1= (pow(-1., n+1)/pnl_sf_fact(n)* n/2.*strike*pow(V1+V2, n/2.-1.)*pow(y
    *Cn3V2= (pow(-1., n+1)/pnl_sf_fact(n)* n/2.*strike*pow(V1+V2, n/2.-1.)*pow(y
}
void Cal_C4(int n,  double y, double V1, double V2, double strike, double cdf, d
{
    *Cn4 = (pow(-1., n+1)/pnl_sf_fact(n)* strike *pow(V1+V2, n/2.)*pow(y,n)-pn1*
}
```

```
static void BarrierPrix(int N, double y, double tau, double strike,   double S0,
{

    int n;
    double D0,       D1,       D2,       D3,       D4,       D5,       D6,

    double C1, C1V1, C1V1V1, C1V1V1V1, C2, C2V1, C2V2, C2V1V1, C2V2V2, C2V1V2, C
    double the, cdf, pdf, pn0, qn0, pn1, pn1V1, pn1V2, pn1V1V1, pn1V2V2, pn1V1V2
    PnlMat *fni, *gni, *hni, *kni, *hniV1,  *kniV1, *hniV2,  *kniV2,  *hniV1V1,
    PnlVect *PPn, *CCn;

    fni = pnl_mat_create_from_zero (N+1, N+1);
    gni = pnl_mat_create_from_zero (N+1, N+1);
    hni = pnl_mat_create_from_zero (N+1, N+1);
    kni = pnl_mat_create_from_zero (N+1, N+1);
    hniV1 = pnl_mat_create_from_zero (N+1, N+1);
    kniV1 = pnl_mat_create_from_zero (N+1, N+1);
    hniV2 = pnl_mat_create_from_zero (N+1, N+1);
    kniV2 = pnl_mat_create_from_zero (N+1, N+1);
    hniV1V1 = pnl_mat_create_from_zero (N+1, N+1);
    kniV1V1 = pnl_mat_create_from_zero (N+1, N+1);
    hniV2V2 = pnl_mat_create_from_zero (N+1, N+1);
    kniV2V2 = pnl_mat_create_from_zero (N+1, N+1);
    hniV1V2 = pnl_mat_create_from_zero (N+1, N+1);
    kniV1V2 = pnl_mat_create_from_zero (N+1, N+1);
    hniV2V1 = pnl_mat_create_from_zero (N+1, N+1);
    kniV2V1 = pnl_mat_create_from_zero (N+1, N+1);
    PPn = pnl_vect_create_from_zero(N+1);
    CCn = pnl_vect_create_from_zero(N+1);


    the = log(strike/S0)/sqrt((V1+V2)*tau);

    Cal_CdfPdf(y, &cdf, &pdf);

    Cal_fg(N, fni, gni);
    Cal_polynomials(y, 1, N, fni, gni, hni, kni, hniV1, kniV1, hniV2, kniV2, hni
    Cal_C1(1, y, V1, V2, strike, pn0, qn0, cdf, pdf, &C1, &C1V1,  &C1V1V1,  &C1V

    pnl_mat_set(hni, 2, 0, (A1*C1 +A21*B1*C1V1 + A22*B1*C1V1 + A21*B2* C1 + A22*
```

```
pnl_mat_set(hniV1, 2, 0, (A1V1*C1 +A1*C1V1 +A21V1*B1*C1V1+A21*B1V1*C1V1+A21*
pnl_mat_set(hniV2, 2, 0, (A1V2*C1 +A1*C1V1 +A21*B1V1*C1V1+A21*B1*C1V1V1 + A2
pnl_mat_set(hniV1V1, 2, 0, (A1V1V1*C1 +A1V1*C1V1 +A1V1*C1V1 +A1*C1V1V1 +A21V
pnl_mat_set(hniV2V2, 2, 0, (A1V2V2*C1 +A1*C1V1V1 +A21*B1V1V1*C1V1 +A21*B1V1*
pnl_mat_set(hniV1V2, 2, 0, (A1V1V1*C1 +A1V1*C1V1 +A1V2*C1V1  +A1*C1V1V1 +A21
pnl_mat_set(hniV2V1, 2, 0, (A1V1V1*C1 +A1V2*C1V1 +A1V1*C1V1 +A1*C1V1V1 +A21V

Cal_polynomials(y, 2, N, fni, gni, hni, kni, hniV1, kniV1, hniV2, kniV2, hni
Cal_C2(2,  y, V1, V2, strike, cdf, pdf, pn0, qn0, pn1, qn1, pn1V1, qn1V1, pn

Cal_CoefD(V1, V2,  dividend,  r,  rho1,  rho2,  sigma1,  sigma2,  theta1,  t
pnl_mat_set(hni, 3, 0, A1*C2+ A21*B1*C2V1 +A22*B1*C2V2+ 2.*A21*B2*C2 + 2.*A2
pnl_mat_set(hniV1, 3, 0, A1V1*C2+ A1*C2V1 +A21V1*B1*C2V1 +A21*B1V1*C2V1 +A21
pnl_mat_set(hniV2, 3, 0, A1V2*C2+ A1*C2V2 +A21*B1V1*C2V1 +A21*B1*C2V1V2+ A22
pnl_mat_set(kni, 3, 0, (2.*pnl_mat_get(hni, 3,0) +D1*pnl_mat_get(fni, 2,1) +
pnl_mat_set(kniV1, 3, 0, (2.*pnl_mat_get(hniV1, 3,0) +D1V1*pnl_mat_get(fni,
pnl_mat_set(kniV2, 3, 0, (2.*pnl_mat_get(hniV2, 3,0) +D1V2*pnl_mat_get(fni,
Cal_polynomials(y, 3, N, fni, gni, hni, kni, hniV1, kniV1, hniV2, kniV2, hni
Cal_C3(3,  y, V1, V2, strike, cdf, pdf, pn0, qn0, pn1, qn1, pn1V1, qn1V1, pn

Cal_CoefD(V1, V2,  dividend,  r,  rho1,  rho2,  sigma1,  sigma2,  theta1,  t

pnl_mat_set(hni, 4, 0, (A1*C3 +A21*B1*C3V1 +A22*B1*C3V2 +3.*A21*B2*C3 +3.*A2
pnl_mat_set(kni, 4, 0, (4.*pnl_mat_get(hni, 4,0)+ D1*pnl_mat_get(fni, 3,1) +
pnl_mat_set(hni, 4, 1, 2.*pnl_mat_get(hni, 4, 0) + A21*B1*pnl_mat_get(hniV1,
Cal_polynomials(y, 4, N, fni, gni, hni, kni, hniV1, kniV1, hniV2, kniV2, hni
Cal_C4(4,  y, V1, V2, strike, cdf, pdf, pn0, qn0, pn1, qn1, &C4);


Cal_CdfPdf(the, &cdf, &pdf);
Cal_fg(N, fni, gni);
Cal_polynomials(the, 1, N, fni, gni, hni, kni, hniV1, kniV1, hniV2, kniV2, h
pnl_vect_set(PPn, 1, C1*(pn0*cdf+qn0*pdf)+pn1*cdf+qn1*pdf);
Cal_polynomials(the, 2, N, fni, gni, hni, kni, hniV1, kniV1, hniV2, kniV2, h
pnl_vect_set(PPn, 2, C2*(pn0*cdf+qn0*pdf)+pn1*cdf+qn1*pdf);
Cal_polynomials(the, 3, N, fni, gni, hni, kni, hniV1, kniV1, hniV2, kniV2, h
pnl_vect_set(PPn, 3, C3*(pn0*cdf+qn0*pdf)+pn1*cdf+qn1*pdf);
Cal_polynomials(the, 4, N, fni, gni, hni, kni, hniV1, kniV1, hniV2, kniV2, h
pnl_vect_set(PPn, 4, C4*(pn0*cdf+qn0*pdf)+pn1*cdf+qn1*pdf);

sum =0.0;
```

```
    for (n=1; n<N-1; n++)
    {
        sum= sum + pnl_vect_get(PPn, n)* pow(tau, n/2.);
    }
    *BarPrix = sum;


    pnl_mat_free(&fni);
    pnl_mat_free(&gni);
    pnl_mat_free(&hni);
    pnl_mat_free(&kni);
    pnl_mat_free(&hniV1);
    pnl_mat_free(&kniV1);
    pnl_mat_free(&hniV2);
    pnl_mat_free(&kniV2);
    pnl_mat_free(&hniV1V1);
    pnl_mat_free(&kniV1V1);
    pnl_mat_free(&hniV2V2);
    pnl_mat_free(&kniV2V2);
    pnl_mat_free(&hniV1V2);
    pnl_mat_free(&kniV1V2);
    pnl_mat_free(&hniV2V1);
    pnl_mat_free(&kniV2V1);
    pnl_vect_free(&PPn);

}
static void EuropeanPrix(double L, int Np, double tau, double strike,  double S0
{
    double a, b, c1, c2, u, sum;
    dcomplex h1, h2, d1, d2, w1, w2;
    int j;
    PnlVectComplex *phi, *exi;
    PnlVect *Uj;

    phi = pnl_vect_complex_create(Np);
    exi = pnl_vect_complex_create(Np);
    Uj  = pnl_vect_create(Np);
    c1 =  (-theta1*tau/2. + (theta1-V1)*(1.- exp(-k1*tau))/2./k1) +(r-dividend)*
    c2 = tau*theta1 - (theta1-V1+theta1*sigma1*rho1*tau)/k1 + (sigma1*rho1*(2.*t
    a = c1 - L*sqrt(fabs(c2));
    b = c1 + L*sqrt(fabs(c2));
```

```
    sum =0.0;
    for (j=0; j<Np; j++)
    {
        u = j*M_PI/(b-a);
        d1 = Csqrt(Cadd(Cpow_real(RCsub(k1,  RCmul(rho1*sigma1*u, CI)), 2.0) , R
        d2 = Csqrt(Cadd(Cpow_real(RCsub(k2,  RCmul(rho2*sigma2*u, CI)), 2.0) , R
        h1 = Cdiv( RCsub(k1, Cadd(RCmul(rho1*sigma1*u, CI), d1)), RCsub(k1, Csub
        h2 = Cdiv( RCsub(k2, Cadd(RCmul(rho2*sigma2*u, CI), d2)), RCsub(k2, Csub
        w1 = Cadd(RCmul (k1*theta1/pow(sigma1, 2.), Csub(CRmul( RCsub(k1, Cadd(R
        w2 = Cadd(RCmul (k2*theta2/pow(sigma2, 2.), Csub(CRmul( RCsub(k2, Cadd(R
        pnl_vect_complex_set(phi, j, Cexp( Cadd(RCmul((r-dividend)*u*tau , CI),
        pnl_vect_complex_set(exi, j, Cexp( CRmul(CI, u*(log(S0/strike)-a))) );
        pnl_vect_set(Uj, j, sin(-u*a)/u - (cos(-u*a)- exp(a) + u* sin(-u*a))/(1.
        if (j==0)
        {
            pnl_vect_complex_set(phi, j, RCmul(0.5, pnl_vect_complex_get(phi, j)
            pnl_vect_set(Uj, j, -a - (cos(-u*a)- exp(a) + u* sin(-u*a))/(1.+pow(

        }

        sum = sum + Creal(Cmul(pnl_vect_complex_get(phi, j) , pnl_vect_complex_g
    }
    * EuroPrix = 2./(b-a)* exp(-(r-dividend)*tau)*strike*sum;
    pnl_vect_complex_free(&phi);
    pnl_vect_complex_free(&exi);
}

static void EuropeanPrix2(int N, double tau, double strike,  double S0, double V
{
    int n;
    double D0,       D1,       D2,       D3,       D4,       D5,       D6,
    double C1, C1V1, C1V1V1, C1V1V1V1, C2, C2V1, C2V2, C2V1V1, C2V2V2, C2V1V2, C
    double theta, cdf, pdf, pn0, qn0, pn1, pn1V1, pn1V2, pn1V1V1, pn1V2V2, pn1V1
    PnlMat *fni, *gni, *hni, *kni, *hniV1,  *kniV1, *hniV2,  *kniV2,  *hniV1V1,
    PnlVect *EPn, *CCn;

    fni = pnl_mat_create_from_zero (N+1, N+1);
    gni = pnl_mat_create_from_zero (N+1, N+1);
    hni = pnl_mat_create_from_zero (N+1, N+1);
    kni = pnl_mat_create_from_zero (N+1, N+1);
```

```
hniV1 = pnl_mat_create_from_zero (N+1, N+1);
kniV1 = pnl_mat_create_from_zero (N+1, N+1);
hniV2 = pnl_mat_create_from_zero (N+1, N+1);
kniV2 = pnl_mat_create_from_zero (N+1, N+1);
hniV1V1 = pnl_mat_create_from_zero (N+1, N+1);
kniV1V1 = pnl_mat_create_from_zero (N+1, N+1);
hniV2V2 = pnl_mat_create_from_zero (N+1, N+1);
kniV2V2 = pnl_mat_create_from_zero (N+1, N+1);
hniV1V2 = pnl_mat_create_from_zero (N+1, N+1);
kniV1V2 = pnl_mat_create_from_zero (N+1, N+1);
hniV2V1 = pnl_mat_create_from_zero (N+1, N+1);
kniV2V1 = pnl_mat_create_from_zero (N+1, N+1);
EPn = pnl_vect_create_from_zero(N+1);
CCn = pnl_vect_create_from_zero(N+1);

theta = log(strike/S0)/sqrt((V1+V2)*tau);

Cal_fg(N, fni, gni);


C1 = strike * sqrt(V1+V2);
C2 = - strike *(V1+V2)/2.;
C3 = strike * pow(V1+V2, 3./2)/(3.*2.* pnl_mat_get(fni, 3, 0));
C4 = - strike * pow(V1+V2, 2. )/(4.*3.*2.* pnl_mat_get(fni, 4, 0));

Cal_CdfPdf(theta, &cdf, &pdf);

Cal_polynomials(theta, 1, N, fni, gni, hni, kni, hniV1, kniV1, hniV2, kniV2,
pnl_vect_set(EPn, 1, C1*(pn0*cdf+qn0*pdf)+pn1*cdf+qn1*pdf);

C1V1 = strike* pow(V1+V2, -0.5)/2.;
C1V1V1 = - strike * pow(V1+V2, -3./2)/4.;
C1V1V1V1 = strike * pow(V1+V2, -5./2)*3./2/4.;
pnl_mat_set(hni, 2, 0, (A1*C1 +A21*B1*C1V1 + A22*B1*C1V1 + A21*B2* C1 + A22*
pnl_mat_set(hniV1, 2, 0, (A1V1*C1 +A1*C1V1 +A21V1*B1*C1V1+A21*B1V1*C1V1+A21*
pnl_mat_set(hniV2, 2, 0, (A1V2*C1 +A1*C1V1 +A21*B1V1*C1V1+A21*B1*C1V1V1 + A2
pnl_mat_set(hniV1V1, 2, 0, (A1V1V1*C1 +A1V1*C1V1 +A1V1*C1V1 +A1*C1V1V1 +A21V
pnl_mat_set(hniV2V2, 2, 0, (A1V2V2*C1 +A1*C1V1V1 +A21*B1V1V1*C1V1 +A21*B1V1*
pnl_mat_set(hniV1V2, 2, 0, (A1V1V1*C1 +A1V1*C1V1 +A1V2*C1V1  +A1*C1V1V1 +A21
pnl_mat_set(hniV2V1, 2, 0, (A1V1V1*C1 +A1V2*C1V1 +A1V1*C1V1 +A1*C1V1V1 +A21V
Cal_polynomials(theta, 2, N, fni, gni, hni, kni, hniV1, kniV1, hniV2, kniV2,
```

```
pnl_vect_set(EPn, 2, C2*(pn0*cdf+qn0*pdf)+pn1*cdf+qn1*pdf);

C2V1 = - strike/2.;
C2V2 = - strike/2.;
C2V1V1 = 0.;
C2V2V1 = 0.;
C2V2V2 = 0.;
C2V1V2 = 0.;
Cal_CoefD(V1, V2, dividend, r, rho1, rho2, sigma1, sigma2, theta1, t
pnl_mat_set(hni, 3, 0, A1*C2+ A21*B1*C2V1 +A22*B1*C2V2+ 2.*A21*B2*C2 + 2.*A2
pnl_mat_set(hniV1, 3, 0, A1V1*C2+ A1*C2V1 +A21V1*B1*C2V1 +A21*B1V1*C2V1 +A21
pnl_mat_set(hniV2, 3, 0, A1V2*C2+ A1*C2V2 +A21*B1V1*C2V1 +A21*B1*C2V1V2+ A22
pnl_mat_set(kni, 3, 0, (2.*pnl_mat_get(hni, 3,0) +D1*pnl_mat_get(fni, 2,1) +
pnl_mat_set(kniV1, 3, 0, (2.*pnl_mat_get(hniV1, 3,0) +D1V1*pnl_mat_get(fni,
pnl_mat_set(kniV2, 3, 0, (2.*pnl_mat_get(hniV2, 3,0) +D1V2*pnl_mat_get(fni,
Cal_polynomials(theta, 3, N, fni, gni, hni, kni, hniV1, kniV1, hniV2, kniV2,
pnl_vect_set(EPn, 3, C3*(pn0*cdf+qn0*pdf)+pn1*cdf+qn1*pdf);

C3V1 = strike*pow(V1+V2, 1./2.)*3./2./(3.*2.*pnl_mat_get(fni, 3, 0));
C3V2 = strike*pow(V1+V2, 1./2.)*3./2./(3.*2.*pnl_mat_get(fni, 3, 0));
Cal_CoefD(V1, V2, dividend, r, rho1, rho2, sigma1, sigma2, theta1, t
pnl_mat_set(hni, 4, 0, (A1*C3 +A21*B1*C3V1 +A22*B1*C3V2 +3.*A21*B2*C3 +3.*A2
pnl_mat_set(kni, 4, 0, (4.*pnl_mat_get(hni, 4,0)+ D1*pnl_mat_get(fni, 3,1) +
pnl_mat_set(hni, 4, 1, 2.*pnl_mat_get(hni, 4, 0) + A21*B1*pnl_mat_get(hniV1,

Cal_polynomials(theta, 4, N, fni, gni, hni, kni, hniV1, kniV1, hniV2, kniV2,
pnl_vect_set(EPn, 4, C4*(pn0*cdf+qn0*pdf)+pn1*cdf+qn1*pdf);


sum =0.0;
for (n=1; n<N-1; n++)// expansion of (N-2)-terms;
{
    sum= sum + pnl_vect_get(EPn, n)* pow(tau, n/2.);
}
*EuroPrix2 = sum;

pnl_mat_free(&fni);
pnl_mat_free(&gni);
pnl_mat_free(&hni);
pnl_mat_free(&kni);
pnl_mat_free(&hniV1);
```

```
    pnl_mat_free(&kniV1);
    pnl_mat_free(&hniV2);
    pnl_mat_free(&kniV2);
    pnl_mat_free(&hniV1V1);
    pnl_mat_free(&kniV1V1);
    pnl_mat_free(&hniV2V2);
    pnl_mat_free(&kniV2V2);
    pnl_mat_free(&hniV1V2);
    pnl_mat_free(&kniV1V2);
    pnl_mat_free(&hniV2V1);
    pnl_mat_free(&kniV2V1);
    pnl_vect_free(&EPn);
}

static void American_put_DHeston(double tau, double strike,  double S0, double V
{
    int dir, search, i, N;
    double A1,  A21, A22, A31,  A32,  A41,  A42,  B1, B2, B4, B6, A1V1, A1V2, A1
    double theta, step, y,  price0, price1, price2, BarPrice, EuroPrice1, EuroPr

    N=4;
    theta= log(strike/S0)/(sqrt(V1+V2)*sqrt(tau));
    Cal_CoefAB(V1, V2, dividend, r, rho1, rho2, sigma1, sigma2, theta1,  theta2,

    step = 1.;
    y= theta;
    price0=strike-S0;

    while (step>=0.001)
    {
        dir=1;
        search=1;
        i=0;
        while (search==1 && i<=1000)
        {
            i=i+1;
            BarrierPrix( N, y+dir*step, tau, strike, S0, V1, V2, dividend, r, rh

            if (price1>price0)
            {
                y=y+dir*step;
```

```
                price0=price1;
            }
            else
            {
                search=0;
            }
        }
        step=step/10;
        BarrierPrix( N, y-step, tau, strike, S0, V1, V2, dividend, r, rho1, rho2
        BarrierPrix( N, y+step, tau, strike, S0, V1, V2, dividend, r, rho1, rho2

        if (price2>price1) dir=1;
        else dir=-1;
    }

    BarPrice = price0; // Approximated American put price by Barrier option pric
    EuropeanPrix(10, 80, tau, strike,  S0, V1, V2, dividend, r, rho1, rho2, sigm
    EuropeanPrix2(N, tau, strike,  S0, V1, V2, dividend, r, rho1, rho2, sigma1,

    *EuroPrice = EuroPrice1;
    *AmePrice = EuroPrice1 + BarPrice - EuroPrice2;//True American put price

}

int ApZhengFengDoubleHeston(double s0, NumFunc_1  *p, double T, double r, double
{
  double strike;
  double EuroPrice, AmePrice;

  strike = p->Par[0].Val.V_PDOUBLE;
  American_put_DHeston(T,  strike, s0,  V1,  V2,  dividend,  r,  rho1,  rho2,  s
  *ptprice =AmePrice;


  return OK;
}
int CALC(AP_ZhengFeng_DoubleHeston)(void *Opt, void *Mod, PricingMethod *Met)
{
  TYPEOPT *ptOpt = (TYPEOPT *)Opt;

  TYPEMOD *ptMod = (TYPEMOD *)Mod;
```

```
  double r, divid;

  if (ptMod->Sigma.Val.V_PDOUBLE == 0.0)
    {
      Fprintf(TOSCREEN, "BLACK-SCHOLES MODEL\ n\ n\ n");
      return WRONG;
    }
  else
    {
      r = log(1. + ptMod->R.Val.V_DOUBLE / 100.);
      divid = log(1. + ptMod->Divid.Val.V_DOUBLE / 100.);

      return ApZhengFengDoubleHeston(ptMod->S0.Val.V_PDOUBLE,
                              ptOpt->PayOff.Val.V_NUMFUNC_1,
                              ptOpt->Maturity.Val.V_DATE - ptMod->T.Val.V_DATE,
                              r, divid, ptMod->Sigma0.Val.V_PDOUBLE,
                              ptMod->Sigma0V.Val.V_PDOUBLE,
                              ptMod->LongRunVariance.Val.V_PDOUBLE,
                              ptMod->LongRunVarianceV.Val.V_PDOUBLE,
                              ptMod->MeanReversion.Val.V_PDOUBLE,
                              ptMod->MeanReversionV.Val.V_PDOUBLE,
                              ptMod->Sigma.Val.V_PDOUBLE,
                              ptMod->SigmaV.Val.V_PDOUBLE,
                              ptMod->Rho.Val.V_DOUBLE,
                              ptMod->RhoSV2.Val.V_DOUBLE,
                              &(Met->Res[0].Val.V_DOUBLE)
                             );
    }
}


static int CHK_OPT(AP_ZhengFeng_DoubleHeston)(void *Opt, void *Mod)
{
  if ((strcmp(((Option *)Opt)->Name, "PutAmer") == 0))
    return OK;

  return  WRONG;
}
#endif //PremiaCurrentVersion

static int MET(Init)(PricingMethod *Met, Option *Opt)
```

```
{
  if (Met->init == 0)
    {
      Met->init = 1;
    }

  return OK;
}

PricingMethod MET(AP_ZhengFeng_DoubleHeston) =
{
  "AP_ZhengFeng_DoubleHeston",
  {{" ", PREMIA_NULLTYPE, {0}, FORBID}},
  CALC(AP_ZhengFeng_DoubleHeston),
  { {"Price", DOUBLE, {100}, FORBID},
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CHK_OPT(AP_ZhengFeng_DoubleHeston),
  CHK_ok,
  MET(Init)
};
```