Help

```
#include "mer2d_std2d.h"

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <complex.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 < (2016+2) //The "#els
static int CHK_OPT(AP_FourierCosine_Merton2d)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(AP_FourierCosine_Merton2d)(void *Opt, void *Mod, PricingMethod *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else


void static range(int L, double r, double s01, double s02, double sigma1, double
{
  double mu1, mu2, m11, m12, m14, m21, m22, m24;

  mu1= (r-lambda*(exp(m1+v1*v1/2.)-1.) - sigma1*sigma1/2)*T/M;
  mu2= (r-lambda*(exp(m2+v2*v2/2.)-1.) - sigma2*sigma2/2)*T/M;
  m11 = mu1 + lambda*T/M*m1;
  m12 = sigma1*sigma1*T/M+lambda*T/M*(m1*m1+v1*v1);
  m14 = lambda*T/M* ( pow(m1,4) + 6*v1*v1*m1*m1 + 3*pow(v1,4));
  m21 = mu2 + lambda*T/M*m2;
  m22 = sigma2*sigma2*T/M+lambda*T/M*(m2*m2+v2*v2);
  m24 = lambda*T/M* ( pow(m2,4) + 6*v2*v2*m2*m2 + 3*pow(v2,4));
```

```
  *a1 = log(s01)+m11-L*sqrt(m12+sqrt(m14));
  *b1 = log(s01)+m11+L*sqrt(m12+sqrt(m14));;
  *a2 = log(s02)+m21-L*sqrt(m22+sqrt(m24));;
  *b2 = log(s02)+m21+L*sqrt(m22+sqrt(m24));;
}
void static fphi(double r, double sigma1, double sigma2, double lambda, double m
{
  int i, j;
  double u1, u2, mu1, mu2, xi1, xi2;

  for (i=0; i<N1; i++)
    {
  for (j=0; j<N2; j++)
       {
  u1 = i*M_PI/(b1-a1);
  u2 = j*M_PI/(b2-a2);
  xi1= exp(m1+v1*v1/2.)-1.;
  xi2= exp(m2+v2*v2/2.)-1.;
  mu1= (r-lambda*xi1 - sigma1*sigma1/2)*T/M;
  mu2= (r-lambda*xi2 - sigma2*sigma2/2)*T/M;
  pnl_mat_complex_set(chf1, i, j, Cexp(Cadd ( Complex( -0.5*(pow(u1*sigma1, 2)*T
  pnl_mat_complex_set(chf2, i, j, Cexp(Cadd ( Complex( -0.5*(pow(u1*sigma1, 2)*T

  pnl_mat_complex_set(Fplus, i, j, RCmul(1., Cmul(pnl_mat_complex_get(chf1, i, j
  pnl_mat_complex_set(Fminu, i, j, RCmul(1., Cmul(pnl_mat_complex_get(chf2, i, j

       }
    }
}
void static vfft(int N1, int N2, int N, double K, double a1, double b1, double a
{
  int j1,j2;
  double x1, x2;
  PnlMat *vin=pnl_mat_create_from_zero(2*N1, 2*N2);
  PnlMatComplex *vout=pnl_mat_complex_create_from_zero(2*N1, 2*N2);

  //call option
  if (flag==1)
    {
  //the case of j1=0, j2=0 & N2
  x1=a1;
```

```
x2=a2;
pnl_mat_set(vin, 0, 0, 4.*MAX(exp(x1)/2.+exp(x2)/2.-K, 0));
pnl_mat_set(vin, 0, N2, 0);

//the case of j1=0, j2=[1, N2-1]&[N2+1, 2*N2-1]
for (j2=1; j2<N2; j2++)
      {
x1=a1;
x2=a2+(b2-a2)/N2*j2;
pnl_mat_set(vin, 0, j2, 2.*MAX(exp(x1)/2.+exp(x2)/2.-K, 0));
pnl_mat_set(vin, 0, 2*N2-j2, 2.*MAX(exp(x1)/2.+exp(x2)/2.-K, 0));
      }

// for j1=[1, N1-1] &[N1+1, 2*N1-1],  j2=0 & N2 & [1, N2-1] &[N2+1, 2*N2-1]
for (j1=1; j1<N1; j1++)
      {
x1=a1+(b1-a1)/N1*j1;
// j1=[1, N1-1] & [N1+1, 2*N1-1], j2=0 & N2
x2=a2;
pnl_mat_set(vin, j1, 0, 2.*MAX(exp(x1)/2.+exp(x2)/2.-K, 0));
pnl_mat_set(vin, 2*N1-j1, 0, 2.*MAX(exp(x1)/2.+exp(x2)/2.-K, 0));
pnl_mat_set(vin, j1, N2, 0);
pnl_mat_set(vin, 2*N1-j1, N2, 0);
// j1=[1, N1-1] &[N1+1, 2*N1-1],  j2=[1, N2-1] &[N2+1, 2*N2-1]
for (j2=1; j2<N2; j2++)
          {
x2=a2+(b2-a2)/N2*j2;
pnl_mat_set(vin, j1, j2, MAX(exp(x1)/2.+exp(x2)/2.-K, 0));
pnl_mat_set(vin, j1, 2*N2-j2, MAX(exp(x1)/2.+exp(x2)/2.-K, 0));
pnl_mat_set(vin, 2*N1-j1, j2, MAX(exp(x1)/2.+exp(x2)/2.-K, 0));
pnl_mat_set(vin, 2*N1-j1, 2*N2-j2, MAX(exp(x1)/2.+exp(x2)/2.-K, 0));
          }
      }

  }
else//put option
  {
//the case of j1=0, j2=0 & N2
x1=a1;
x2=a2;
pnl_mat_set(vin, 0, 0, 4.*MAX(K-exp(x1)/2.-exp(x2)/2., 0));
```

```
pnl_mat_set(vin, 0, N2, 0);

//the case of j1=0, j2=[1, N2-1]&[N2+1, 2*N2-1]
for (j2=1; j2<N2; j2++)
      {
x1=a1;
x2=a2+(b2-a2)/N2*j2;
pnl_mat_set(vin, 0, j2, 2.*MAX(K-exp(x1)/2.-exp(x2)/2., 0));
pnl_mat_set(vin, 0, 2*N2-j2, 2.*MAX(K-exp(x1)/2.-exp(x2)/2., 0));
      }

// for j1=[1, N1-1] &[N1+1, 2*N1-1],  j2=0 & N2 & [1, N2-1] &[N2+1, 2*N2-1]
for (j1=1; j1<N1; j1++)
      {
x1=a1+(b1-a1)/N1*j1;
// j1=[1, N1-1] & [N1+1, 2*N1-1], j2=0 & N2
x2=a2;
pnl_mat_set(vin, j1, 0, 2.*MAX(K-exp(x1)/2.-exp(x2)/2., 0));
pnl_mat_set(vin, 2*N1-j1, 0, 2.*MAX(K-exp(x1)/2.-exp(x2)/2., 0));
pnl_mat_set(vin, j1, N2, 0);
pnl_mat_set(vin, 2*N1-j1, N2, 0);
// j1=[1, N1-1] &[N1+1, 2*N1-1],  j2=[1, N2-1] &[N2+1, 2*N2-1]
for (j2=1; j2<N2; j2++)
          {
x2=a2+(b2-a2)/N2*j2;
pnl_mat_set(vin, j1, j2, MAX(K-exp(x1)/2.-exp(x2)/2., 0));
pnl_mat_set(vin, j1, 2*N2-j2, MAX(K-exp(x1)/2.-exp(x2)/2., 0));
pnl_mat_set(vin, 2*N1-j1, j2, MAX(K-exp(x1)/2.-exp(x2)/2., 0));
pnl_mat_set(vin, 2*N1-j1, 2*N2-j2, MAX(K-exp(x1)/2.-exp(x2)/2., 0));
          }
      }
  }

//for j1=N1, j2=0 &N2 &[1,N2-1] &[2*N2-1]
for (j2=0; j2<2*N2-1; j2++)
  {
pnl_mat_set(vin, N1, j2, 0);
  }

pnl_real_fft2d(vin, vout);
```

```
  for (j1=0; j1<N; j1++)
    {
  for (j2=0; j2<N; j2++)
        {
  pnl_mat_set(ve, j1, j2, Creal(pnl_mat_complex_get(vout, j1, j2))/N1/N2);
        }
    }

  pnl_mat_free(&vin);
  pnl_mat_complex_free(&vout);
}
void static ContValue(int N1, int N2, double K, double a1, double b1, double a2,
{
  double sum=0.0;
  int k1, k2;
  dcomplex chfexp;

  chfexp = Cadd(Cmul(pnl_mat_complex_get(chf1, 0, 0), Cexp(Complex(0., 0*M_PI*(x
  sum= pnl_mat_get(ve, 0,0)*Creal(chfexp)/8;
  for (k2=1; k2<N2; k2++)
    {
  chfexp = Cadd(Cmul(pnl_mat_complex_get(chf1, 0, k2), Cexp(Complex(0., 0*M_PI*(
  sum=sum + pnl_mat_get(ve, 0, k2)*Creal(chfexp)/4;
    }
  for (k1=1; k1<N1; k1++)
    {
  chfexp = Cadd(Cmul(pnl_mat_complex_get(chf1, k1, 0), Cexp(Complex(0., k1*M_PI*
  sum=sum + pnl_mat_get(ve, k1, 0)*Creal(chfexp)/4;
  for (k2=1; k2<N2; k2++)
        {
  chfexp = Cadd(Cmul(pnl_mat_complex_get(chf1, k1, k2), Cexp(Complex(0., k1*M_PI
  sum=sum + 0.5*pnl_mat_get(ve, k1, k2)*Creal(chfexp);
        }
    }
  *CValue= sum;
}
void static ExerBound(int N1, int N2, int gJ, int flag, double r, double T, int
{
  int i, j;
  double g1, x, sum1=0;
```

```
  pnl_vect_set_zero(star);
  for (j=0; j<gJ; j++)
    {
  for (i=0; i<gJ; i++)
        {
  x=a1+(b1-a1)*i/gJ;

  ContValue(N1, N2, strike, a1, b1, a2, b2, ve, chf1, chf2, x, a2+(b2-a2)/gJ*j,
  if (flag==1)//call option:
            {
  g1=((exp(x)+exp(a2+(b2-a2)/gJ*j))/2-strike<=0.)?0.: (exp(x)+exp(a2+(b2-a2)/gJ*
  if (g1-exp(-r*T/M)*sum1>=0)
                {
  pnl_vect_set(star, j, x);
  break;
                }
            }
  else//put option
            {
  g1=(strike-(exp(x)+exp(a2+(b2-a2)/gJ*j))/2<=0.)?0.: strike-(exp(x)+exp(a2+(b2-
  if (exp(-r*T/M)*sum1-g1>=0)
                {
  pnl_vect_set(star, j, x);
  break;
                }
            }
        }
    }
}
void static funcbeta(int N1, int N2, double r, double T, int M, PnlMatComplex *c
{
  int k1, k2;
  for (k1=0; k1<N1; k1++)
    {
  for (k2=0; k2<N2; k2++)
        {
  pnl_mat_complex_set(beta0, k1, k2, Complex(0.5*pnl_mat_get(ve, k1, k2), 0.));
  pnl_mat_complex_set(beta1, k1, k2, CRmul(pnl_mat_complex_get(chf1, k1, k2), 0.
  pnl_mat_complex_set(beta2, k1, k2, CRmul(pnl_mat_complex_get(chf2, k1, k2), 0.
  //pnl_mat_complex_set(gbeta, k1, k2, Cadd(Cmul(pnl_mat_complex_get(beta1, k1,
  //printf("k1=%d, k2=%d, beta1=%f, beta2=%f, gbeta=%f\ n", k1,k2,Creal(pnl_mat_
```

```
        }
    }
}
void static mbasic(int Nnumber, double a, double b, double l, double u, PnlVectC
{
  int n;
  pnl_vect_complex_set(mjp, 0, RCmul((u-l)*M_PI/(b-a), CI));
  pnl_vect_complex_set(mjm, 0, RCmul((u-l)*M_PI/(b-a), CI));
  for (n=1;n<2*Nnumber+1;n++)
    {
  pnl_vect_complex_set(mjp, n, CRdiv(Csub(Cexp(RCmul(n*(u-a)*M_PI/(b-a) ,CI)) ,C
  pnl_vect_complex_set(mjm, n, RCmul(-1., Conj(CRdiv(Csub(Cexp(RCmul(n*(u-a)*M_P
    }
}
void static c_fft(int number, PnlVectComplex  *mj, PnlVectComplex *beta, PnlVect
{
  PnlVectComplex *ms=pnl_vect_complex_create_from_zero(2*number);
  PnlVectComplex *mc=pnl_vect_complex_create_from_zero(2*number);
  PnlVectComplex *us=pnl_vect_complex_create_from_zero(2*number);
  PnlVectComplex *ivector1 = pnl_vect_complex_create_from_zero (2*number);
  PnlVectComplex *ivector2 = pnl_vect_complex_create_from_zero (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_vect_complex_get(beta, n));
  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_vect_complex_get(beta, 0)));
  for (n=number; 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_com
  pnl_vect_complex_set(ivector2, n,Cmul(pnl_vect_complex_get(mc, n),pnl_vect_com
    }
  for (n=0;n<number;n++)
    {
  pnl_vect_complex_set(ivector2, 2*n+1, RCmul(-1., pnl_vect_complex_get(ivector2
    }
  pnl_ifft_inplace (ivector1);
  pnl_ifft_inplace (ivector2);
  for(n=0; n<number; n++)
    {
  //pnl_vect_complex_set(result, n, Cmul(Complex(0.,-1./M_PI),Cadd(pnl_vect_comp
  pnl_vect_complex_set(result, n, Cmul(Complex(0.,-1./M_PI),Cadd(pnl_vect_comple
  //pnl_vect_set(result, n, exp(-r*T/M)/M_PI*(Cimag(pnl_vect_complex_get(ivector
    }

  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);
}

void static Vk(int N1, int N2, int gJ, double strike, double a1, double b1, doub
{
  int i, k1, k2, j1;

  PnlMat *gadd = pnl_mat_create_from_zero(N1, N2);
  PnlMat *cadd = pnl_mat_create_from_zero(N1, N2);
  PnlVectComplex *beta= pnl_vect_complex_create_from_zero(N1);
  PnlVectComplex *mjp= pnl_vect_complex_create_from_zero(2*N1+1);
  PnlVectComplex *mjm= pnl_vect_complex_create_from_zero(2*N1+1);
  PnlVectComplex *vk=pnl_vect_complex_create_from_zero(N1);
  PnlVectComplex *vkm=pnl_vect_complex_create_from_zero(N1);
  PnlMatComplex *tempg = pnl_mat_complex_create_from_zero(N1,N2);
  PnlMatComplex *tempc = pnl_mat_complex_create_from_zero(N1,N2);

  pnl_mat_set_zero (ve);
```

```
if (flag==1)
  {
for (i=0; i<gJ; i++)
      {
mbasic(N2, a2, b2, a2+(b2-a2)/gJ*i, a2+(b2-a2)/gJ*(i+1), mjp, mjm);
for (j1=0; j1<N1; j1++)
          {
// Exercise value G on exercise region
pnl_mat_complex_get_row(beta, beta0, j1);
c_fft(N1, mjp, beta, vk);
c_fft(N1, mjm, beta, vkm);
pnl_vect_complex_plus_vect(vk, vkm);
pnl_mat_complex_set_row(tempg, vk, j1);

// Continuation value C on continuation region
pnl_mat_complex_get_row(beta, beta1, j1);
c_fft(N1, mjp, beta, vk);
pnl_mat_complex_get_row(beta, beta2, j1);
c_fft(N1, mjm, beta, vkm);
pnl_vect_complex_plus_vect(vk, vkm);
pnl_mat_complex_set_row(tempc, vk, j1);
          }

// Exercise value G
mbasic(N1, a1, b1, pnl_vect_get(star, i), b1, mjp, mjm);
for (k2=0; k2<N2; k2++)
          {
pnl_mat_complex_get_col(beta, tempg, k2);
c_fft(N1, mjp, beta, vk);
for (k1=0; k1<N1; k1++)
              {
pnl_mat_set(gadd, k1, k2, Creal(pnl_vect_complex_get(vk, k1)));
              }
          }
pnl_mat_plus_mat(ve, gadd);

// Continuation value C
mbasic(N1, a1, b1, a1, pnl_vect_get(star, i), mjp, mjm);
for (k2=0; k2<N2; k2++)
          {
```

```
pnl_mat_complex_get_col(beta, tempc, k2);
c_fft(N1, mjp, beta, vk);
for (k1=0; k1<N1; k1++)
                {
pnl_mat_set(cadd, k1, k2, Creal(pnl_vect_complex_get(vk, k1)));
                }
           }
pnl_mat_plus_mat(ve, cadd);
      }
  }
else
  {
// Exercise value G on exercise region
for (i=0; i<gJ; i++)
      {
mbasic(N2, a2, b2, a2+(b2-a2)/gJ*i, a2+(b2-a2)/gJ*(i+1), mjp, mjm);
for (j1=0; j1<N1; j1++)
           {
// Exercise value G on exercise region
pnl_mat_complex_get_row(beta, beta0, j1);
c_fft(N1, mjp, beta, vk);
c_fft(N1, mjm, beta, vkm);
pnl_vect_complex_plus_vect(vk, vkm);
pnl_mat_complex_set_row(tempg, vk, j1);

// Continuation value C on continuation region
pnl_mat_complex_get_row(beta, beta1, j1);
c_fft(N1, mjp, beta, vk);
pnl_mat_complex_get_row(beta, beta2, j1);
c_fft(N1, mjm, beta, vkm);
pnl_vect_complex_plus_vect(vk, vkm);
pnl_mat_complex_set_row(tempc, vk, j1);
           }

// Exercise value G
mbasic(N1, a1, b1, a1, pnl_vect_get(star, i), mjp, mjm);
for (k2=0; k2<N2; k2++)
           {
pnl_mat_complex_get_col(beta, tempg, k2);
c_fft(N1, mjp, beta, vk);
for (k1=0; k1<N1; k1++)
```

```
                {
  pnl_mat_set(gadd, k1, k2, Creal(pnl_vect_complex_get(vk, k1)));
                }
            }
  pnl_mat_plus_mat(ve, gadd);

  // Continuation value C
  mbasic(N1, a1, b1, pnl_vect_get(star, i), b1, mjp, mjm);
  for (k2=0; k2<N2; k2++)
            {
  pnl_mat_complex_get_col(beta, tempc, k2);
  c_fft(N1, mjp, beta, vk);
  for (k1=0; k1<N1; k1++)
                {
  pnl_mat_set(cadd, k1, k2, Creal(pnl_vect_complex_get(vk, k1)));
                }
            }
  pnl_mat_plus_mat(ve, cadd);
        }
    }
  pnl_mat_free(&gadd);
  pnl_mat_free(&cadd);
  pnl_mat_complex_free(&tempg);
  pnl_mat_complex_free(&tempc);
  pnl_vect_complex_free(&vk);
  pnl_vect_complex_free(&vkm);
  pnl_vect_complex_free(&beta);
  pnl_vect_complex_free(&mjp);
  pnl_vect_complex_free(&mjm);
}

////////////////////////////////////////////////////////////////
static int Ap_FourierCosine_Mer2d(double s01,double s02, NumFunc_2  *p, double T
{
  double strike;
  int flag;
  double sum;
  int k1, L, gJ;
  double a1, a2, b1, b2;

  L=10;
```

```
gJ=160;
PnlMatComplex *chf1 = pnl_mat_complex_create_from_zero(N,N);
PnlMatComplex *chf2 = pnl_mat_complex_create_from_zero(N,N);
PnlMatComplex *Fplus = pnl_mat_complex_create_from_zero(N,N);
PnlMatComplex *Fminu = pnl_mat_complex_create_from_zero(N,N);
PnlMatComplex *beta0 = pnl_mat_complex_create_from_zero(N,N);
PnlMatComplex *beta1 = pnl_mat_complex_create_from_zero(N,N);
PnlMatComplex *beta2 = pnl_mat_complex_create_from_zero(N,N);
PnlMat *ve  = pnl_mat_create_from_zero(N,N);
PnlVect *star = pnl_vect_create_from_zero(gJ);
PnlVectComplex *beta= pnl_vect_complex_create_from_zero(N);
PnlVectComplex *mj= pnl_vect_complex_create_from_zero(2*N+1);
PnlVectComplex *vk=pnl_vect_complex_create_from_zero(N);
PnlMatComplex *temp = pnl_mat_complex_create_from_zero(N,N);

 if ((p->Compute) == &CallBasket_2d)
  flag=1;
 else
    flag=-1;

strike = p->Par[0].Val.V_PDOUBLE;
range(L, r, s01, s02, sigma1, sigma2, lambda, m1, m2, v1, v2, rho1, rho2, T,
a2=MIN(a1,a2);
b2=MAX(b1,b2);
a1=a2;
b1=b2;

vfft(2000, 2000, N, strike, a1, b1, a2, b2, flag, ve);
fphi(r, sigma1, sigma2, lambda, m1, m2, v1, v2, rho1, rho2, s01, s02, a1, b1,

for (k1=1; k1<M; k1++)
  {
   funcbeta(N, N,  r, T, M, chf1, chf2, ve, beta0, beta1, beta2);
   ExerBound(N, N, gJ, flag, r, T, M, a1, a2, b1, b2, strike, chf1, chf2, ve,
   Vk(N, N, gJ, strike, a1, b1, a2, b2, flag, star, beta0,  beta1, beta2, ve);
  }

ContValue(N, N, strike, a1, b1, a2, b2, ve, chf1, chf2, log(s01), log(s02), &s
*ptprice=exp(-r*T/M)*sum;

pnl_mat_free(&ve);
```

```
  pnl_mat_complex_free(&chf1);
  pnl_mat_complex_free(&chf2);
  pnl_mat_complex_free(&Fplus);
  pnl_mat_complex_free(&Fminu);
  pnl_mat_complex_free(&beta0);
  pnl_mat_complex_free(&beta1);
  pnl_mat_complex_free(&beta2);
  pnl_vect_free(&star);
  pnl_vect_complex_free(&vk);
  pnl_vect_complex_free(&beta);
  pnl_vect_complex_free(&mj);
  pnl_mat_complex_free(&temp);

  return OK;

}

int CALC(AP_FourierCosine_Merton2d)(void *Opt, void *Mod, PricingMethod *Met)
{
  TYPEOPT *ptOpt = (TYPEOPT *)Opt;
  TYPEMOD *ptMod = (TYPEMOD *)Mod;
  double r;

  r = log(1. + ptMod->R.Val.V_DOUBLE / 100.);

  return  Ap_FourierCosine_Mer2d(ptMod->S01.Val.V_PDOUBLE, ptMod->S02.Val.V_PDOU
}

static int CHK_OPT(AP_FourierCosine_Merton2d)(void *Opt, void *Mod)
{
  if ((strcmp(((Option *)Opt)->Name, "CallBasketAmer_2d") == 0)
      || (strcmp(((Option *)Opt)->Name, "PutBasketAmer_2d") == 0 ))
    return OK;
  return WRONG;
}

#endif //PremiaCurrentVersion

static int MET(Init)(PricingMethod *Met, Option *Opt)
{
  if (Met->init == 0)
```

```
    {
      Met->init = 1;
  Met->HelpFilenameHint = "ap_fouriercosine_merton2d";
      Met->Par[0].Val.V_INT2 = 40;
  Met->Par[1].Val.V_INT2 = 10;
    }

  return OK;
}

PricingMethod MET(AP_FourierCosine_Merton2d) =
{
  "AP_FourierCosine_Merton2d",
  {{"Number of integration steps", INT2, {100}, ALLOW},{"Number of early-exercis
  CALC(AP_FourierCosine_Merton2d),
  { {"Price", DOUBLE, {100}, FORBID}, {" ", PREMIA_NULLTYPE, {0}, FORBID}},
  CHK_OPT(AP_FourierCosine_Merton2d),
  CHK_ok,
  MET(Init)
} ;
```