Help

```
#include"pnl/pnl_specfun.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"

#include "
href../../../mod/cgmy1d/cgmy1d_pad/cgmy1d_pad_h_src.pdfcgmy1d_pad.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2015+2) //The "#els
static int CHK_OPT(AP_FourierCosine_CGMY_FixedAsian)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(AP_FourierCosine_CGMY_FixedAsian)(void *Opt, void *Mod, PricingMethod *
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else



static void Valomega(int N, double a, double b, PnlVect *omega)
{
  int j;

  for (j = 0; j < N; j++)
    {
      pnl_vect_set(omega, j, ((double)j)*M_PI / (b - a));
    }
}


static void ValExpCos(int nq, double a, double b,  PnlVect *Vexp1, PnlVect *Vcos
{
  int n;
```

```
  double x1, x2;
  for (n = 0; n < nq / 2 + 1; n++)
    {
      x1 = (b - a) / 2 * cos(n * M_PI / nq) + (a + b) / 2;
      x2 = (b - a) / 2 * (-cos(n * M_PI / nq)) + (a + b) / 2;
      pnl_vect_set(Vexp1, n, exp(x1) + 1);
      pnl_vect_set(Vexp2, n, exp(x2) + 1);
      pnl_vect_set(Vcos1, n, (x1 - a));
      pnl_vect_set(Vcos2, n, (x2 - a));
    }
}

static void ValYkl(double uk, double ul, int nq, double a, double b,  PnlVect *V
{
  int n;
  dcomplex fx1, fx2;
  for (n = 0; n < nq / 2 + 1; n++)
    {
      fx1 = RCmul(cos(pnl_vect_get(Vcos1, n) * ul) , Cpow(Complex(pnl_vect_get(V
      fx2 = RCmul(cos(pnl_vect_get(Vcos2, n) * ul) , Cpow(Complex(pnl_vect_get(V
      pnl_vect_complex_set(Ykl, n, RCmul((b - a) / 2, Cadd(fx1, fx2)));
    }
}


static void ValWn(int N, int nq, double a, double b, PnlVect *Wm)
{
  int i;
  PnlVect *dm;
  PnlVectComplex *dft;

  dm = pnl_vect_create(nq);
  dft = pnl_vect_complex_create(nq);

  for (i = 0; i < nq / 2 + 1; i++)
    {
      pnl_vect_set(dm, i, 2 / (1. - (2.*i) * (2.*i)));
    }
  for (i = nq / 2 + 1; i < nq; i++)
    {
      pnl_vect_set(dm, i, pnl_vect_get(dm, nq - i));
```

```
    }
  pnl_real_fft(dm, dft);
  for (i = 0; i < nq / 2 + 1; i++)
    {
      pnl_vect_set(Wm, i, (pnl_vect_complex_get_real(dft, i)) / nq);
    }

  pnl_vect_free(&dm);
  pnl_vect_complex_free(&dft);
}


static void ValMkl(int N, int nq, PnlVect *Wm,  double a, double b, PnlVect *ome
{
  int i, k, l;
  dcomplex sum;
  PnlVectComplex *Ykl;

  Ykl = pnl_vect_complex_create(nq / 2 + 1);
  for (k = 0; k < N; k++)
    {
      for (l = 0; l < N; l++)
        {
          ValYkl(pnl_vect_get(omega, k), pnl_vect_get(omega, l), nq, a, b, Vexp1
          sum = CRmul(pnl_vect_complex_get(Ykl, 0), 0.5 * pnl_vect_get(Wm, 0));
          for (i = 1; i < nq / 2; i++)
            {
              sum = Cadd(sum, CRmul(pnl_vect_complex_get(Ykl, i), pnl_vect_get(W
            }
          sum = Cadd(sum, CRmul(pnl_vect_complex_get(Ykl, nq / 2), 0.5 * pnl_vec
          pnl_mat_complex_set(Mkl, k, l, sum);
        }
    }
  pnl_vect_complex_free(&Ykl);
}

static void ValcfR(int N, double C, double G, double M, double Y, double w, doub
{
  int j;
  double omegaj;
  for (j = 0; j < N; j++)
```

```
    {
      omegaj = pnl_vect_get(omega, j);
      pnl_vect_complex_set(cfR, j, Cexp(Cadd(Complex(-0.5 * pow(omegaj, 2)*pow(s
    }
}


static void ValAj(int N, double a, double b, PnlVectComplex *cfY, PnlVect *Aj)
{
  int l;
  pnl_vect_set(Aj, 0, 0.5 * (pnl_vect_complex_get_real(cfY, 0)*cos(-a * 0 * M_PI
  for (l = 1; l < N; l++)
    {
      pnl_vect_set(Aj, l, pnl_vect_complex_get_real(cfY, l)*cos(-a * l * M_PI /
    }
}


static void ValcfYj(int N, double a, double b, PnlVect *Aj, PnlVectComplex *cfR,
{
  int k, l;
  dcomplex sum;

  pnl_vect_complex_set(cfYj, 0, CONE);
  for (k = 1; k < N; k++)
    {
      sum = CZERO;
      for (l = 0; l < N; l++)
        {
          sum = Cadd(sum, RCmul(pnl_vect_get(Aj, l), pnl_mat_complex_get(Mkl, k,
        }
      sum = RCmul(2. / (b - a), sum);
      pnl_vect_complex_set(cfYj, k, Cmul(pnl_vect_complex_get(cfR, k), sum));
    }
}


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
}
```

```
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
}

void static fV(double a, double b, double S0, double K, int N, int Mn, int flag,
{
  double xstar, xi, psi, result;
  int n;

  result = 0.;
  xstar = log(K * (Mn + 1) / S0 - 1);
  for (n = 0; n < N; n++)
    {
      fxi(a, b, a, xstar, n, &xi);
      fpsi(a, b, a, xstar, n, &psi);
      result = 2. / (b - a) * ((K - S0 / (Mn + 1)) * psi - S0 / (Mn + 1) * xi);
      pnl_vect_set(ve, n, result);
    }
}

int cgmy_ap_fouriercosine_asianfixed(NumFunc_2 *P, double S0, double T, double r
{
  int i, L, flag;
  double w, dt, gamc2, gamc4, gamcf, c1, c2, c4, a, b, sum, sum1, parity;
  double sigma, K;
  PnlVect *omega, *Aj, *dm, *Wm, *ve, *Vexp1, *Vcos1, *Vexp2, *Vcos2;
  PnlVectComplex *cfR, *cfY, *Ykl;
  PnlMatComplex *Mkl;


  K = P->Par[0].Val.V_DOUBLE;

  if ((P->Compute) == &Call_OverSpot2)
    flag = 1;
  //Put Fixed
  else
    flag = 0;

  //Volatility of the diffusion term, treated-case
  sigma = 0.;
```

```
/* Allocation of vector */
omega = pnl_vect_create(N);
Aj  = pnl_vect_create(N);
ve = pnl_vect_create(N);
Vexp1 = pnl_vect_create(nq / 2 + 1);
Vcos1 = pnl_vect_create(nq / 2 + 1);
Vexp2 = pnl_vect_create(nq / 2 + 1);
Vcos2 = pnl_vect_create(nq / 2 + 1);
cfR = pnl_vect_complex_create(N);
cfY = pnl_vect_complex_create(N);
dm  = pnl_vect_create(nq);
Wm  = pnl_vect_create(nq);
Ykl = pnl_vect_complex_create(nq);
Mkl = pnl_mat_complex_create(N, N);

/*Cumulants*/
dt = T / Mn;
gamc2  = pnl_tgamma(2 - Y);
gamc4  = pnl_tgamma(4 - Y);;
gamcf = pnl_tgamma(-Y);
c1 = (r - q) * dt + C * dt * gamcf * Y * (pow(G, Y - 1) - pow(M, Y - 1));
c2 = pow(sigma, 2) * dt + C * dt * gamc2 * (pow(M, Y - 2) + pow(G, Y - 2));
c4 = C * dt * gamc4 * (pow(M, (Y - 4)) + pow(G, (Y - 4)));
w = -C * gamcf * (pow(M - 1, Y) - pow(M, Y) + pow(G + 1, Y) - pow(G, Y));
/*Truncation range*/
L = 7;
a = c1 - L * pow(c2 + pow(c4, 0.5), 0.5);
b = log(Mn) + Mn * c1 + L * pow(Mn * c2 + pow(Mn * c4, 0.5), 0.5);
Valomega(N, a, b, omega);

ValWn(N, nq, a, b, Wm);
ValExpCos(nq, a, b, Vexp1, Vcos1, Vexp2, Vcos2);
ValMkl(N, nq, Wm,  a, b, omega, Vexp1, Vcos1, Vexp2, Vcos2, Mkl);
/*Characteristic function of CGMY model*/
ValcfR(N, C, G, M, Y, w, r, sigma, q, dt, omega, gamcf, cfR);
ValAj(N, a, b, cfR, Aj);

/* Recover the Characteristic Function of YM from that of Y1 */
for (i = 1; i < Mn; i++)
  {
```

```
    ValcfYj(N, a, b, Aj, cfR, Mkl, cfY);
    ValAj(N, a, b, cfY, Aj);
  }

fV(a, b, S0, K, N, Mn,  flag, ve);
sum = pnl_vect_get(Aj, 0) * pnl_vect_get(ve, 0);
for (i = 1; i < N; i++)
  {
    sum = sum + pnl_vect_get(Aj, i) * pnl_vect_get(ve, i);
  }
if (flag == 1)
  {
    sum = exp(-r * T) * sum;
    sum1 = 0;
    for (i = 0; i < Mn + 1; i++)
      {
        sum1 = sum1 + exp(r * i * dt);
      }
    parity = S0 * exp(-r * T) / (Mn + 1) * sum1 - K * exp(-r * T) + sum;
  }
else
  {
    parity = exp(-r * T) * sum;

  }

*ptprice = parity;

//Memory free
pnl_vect_free(&omega);
pnl_vect_free(&Aj);
pnl_vect_free(&ve);
pnl_vect_free(&Vexp1);
pnl_vect_free(&Vcos1);
pnl_vect_free(&Vexp2);
pnl_vect_free(&Vcos2);
pnl_vect_free(&dm);
pnl_vect_free(&Wm);
pnl_vect_complex_free(&cfR);
pnl_vect_complex_free(&cfY);
pnl_vect_complex_free(&Ykl);
```

```
  pnl_mat_complex_free(&Mkl);

  return OK;
}

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

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

  return  cgmy_ap_fouriercosine_asianfixed(ptOpt->PayOff.Val.V_NUMFUNC_2, ptMod-
}

static int CHK_OPT(AP_FourierCosine_CGMY_FixedAsian)(void *Opt, void *Mod)
{
  if ((strcmp(((Option *)Opt)->Name, "AsianCallFixedEuro") == 0) || (strcmp(((Op
    return OK;

  return WRONG;
}

#endif //PremiaCurrentVersion
static int MET(Init)(PricingMethod *Met, Option *Mod)
{
  if (Met->init == 0)
    {
      Met->init = 1;
      Met->HelpFilenameHint = "ap_fouriercosine_cgmy_asianfixed";
      Met->Par[0].Val.V_PINT = 256;
      Met->Par[1].Val.V_PINT = 52;
      Met->Par[2].Val.V_PINT = 400;
    }
  return OK;
}

PricingMethod MET(AP_FourierCosine_CGMY_FixedAsian) =
{
```

```
  "AP_FourierCosine_CGMY_FixedAsian",
  {{"N", INT, {100}, ALLOW}, {"Mn", INT, {100}, ALLOW}, {"Nq", INT, {100}, ALLOW
  CALC(AP_FourierCosine_CGMY_FixedAsian),
  {{"Price", DOUBLE, {100}, FORBID}, {" ", PREMIA_NULLTYPE, {0}, FORBID}},
  CHK_OPT(AP_FourierCosine_CGMY_FixedAsian),
  CHK_ok,
  MET(Init)
} ;
```