Help

```
#include <pnl/pnl_mathtools.h>
#include <pnl/pnl_complex.h>
#include"pnl/pnl_specfun.h"
#include "pnl/pnl_integration.h"
#include "
href../../../mod/cgmy1d/cgmy1d_std/cgmy1d_std_h_src.pdfcgmy1d_std.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2016+2) //The "#els

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


typedef struct params_cgmy {
double C;
double G ;
double M ;
double Y;
double mu;
double T;
long int N ;
double K;
double S;
    double x;
double alp;
double eps;
double om;
double d;
double zet;


} params_cgmy;
```

```
static void Set_params(params_cgmy *p, double r, double divid, double Y, double
{

p->C = C;    //in the paper C=c, Y=nu, G=lambda_+, M=-lambda_-.
p->G = G; //lm=lambda_+
p->M = M; //lp=-lambda_-
p->Y = Y;
p->T = t;
p->N = N;
p->K = k;
p->S = s;
p->mu = r-divid+pnl_sf_gamma(-Y)*C*( exp(Y*log(M))-exp(Y*log(M-1)))+pnl_sf_gamma
    p->x = log(s/k)+(p->mu)*t;
    p->alp=MIN(2.0,1./Y+1.);
p->eps=2*M_PI*eps*exp(r*t)/k;
if (ifCall)//Call
    {
p->om=-0.5*M-0.5;
p->d=0.5*(M-1);
}
else
{
p->om=0.5*G;
p->d=0.5*G;
}
p->zet=-M_PI*(p->d)/(1.1*log(p->eps));
//printf("mu= %f\ n",p->mu);
}

static void psiCGMY(params_cgmy *p, dcomplex u, dcomplex *res)//computation for
{
dcomplex aux1, aux2, aux3, aux4;

aux1.r = p->M + u.i;
aux1.i = -u.r;

aux2.r = p->G - u.i;
aux2.i = u.r;
```

```
aux3=Cexp(RCmul(p->Y,Clog(aux1)));
aux4=Cexp(RCmul(p->Y,Clog(aux2)));

res->r = p->C*pnl_sf_gamma(-p->Y)*( exp(p->Y*log(p->M))-aux3.r)+p->C*pnl_sf_gamm
res->i = -p->C*pnl_sf_gamma(-p->Y)*(aux3.i + aux4.i);
}



static dcomplex chip(double eta, double al, double om, double lp)
{
    dcomplex z, aux1, aux2, res;
z.i=lp;
z.r=0;
aux1.i=eta;
aux1.r=lp-om;
aux2=Cexp(RCmul(al,Clog(aux1)));

aux1.i=-pow(lp,1-al);
aux1.r=0.;
aux2=Cmul(aux1,aux2);
res=Cadd(z,aux2);

  return res;
  }

static dcomplex chim(double eta, double al, double om, double lm)
{
    dcomplex z, aux1, aux2, res;
z.i=-lm;
z.r=0;
aux1.i=-eta;
aux1.r=lm+om;
aux2=Cexp(RCmul(al,Clog(aux1)));

aux1.i=pow(lm-1.,1-al);
aux1.r=0.;
aux2=Cmul(aux1,aux2);
res=Cadd(z,aux2);

  return res;
```

```
  }

static double RePutInt(double eta,  void *pp) ///Re( exp(ixChi_+(eta+i om)-T psi
{

params_cgmy *p = (params_cgmy*)pp;

dcomplex ix, aux, aux1, aux2, aux3;

ix.i=p->x;
ix.r=0.;
//////////////exponent computation////////////////////////
aux=chip(eta,p->alp, p->om, p->G);
psiCGMY(p, aux, &aux1);
aux2=Cmul(aux, ix);
aux1=CRmul(aux1,-p->T);
aux1=Cadd(aux1,aux2);
/////////////////end////////////////////////
/////////fraction computation//////////////////
aux2 =Cdiv(Cexp(aux1),aux);
aux.i=aux.i+1.;
aux3 =Cdiv(aux2,aux);
/////////////////end////////////////////////
////////////chi derivative computation/////////////
aux.i=eta/(p->G);
aux.r=1-(p->om)/(p->G);
aux1=Cexp(RCmul(p->alp-1,Clog(aux)));
/////////////////end//////////////////////////////
aux2=Cmul(aux3,aux1);
return aux2.r*(p->alp);
}

static double ReCallInt(double eta,  void *pp) ///Re( exp(ixChi_-(eta+i om)-T ps
{

params_cgmy *p = (params_cgmy*)pp;

dcomplex ix, aux, aux1, aux2, aux3;

ix.i=p->x;
ix.r=0.;
```

```
//////////////exponent computation////////////////////////
aux=chim(eta,p->alp, p->om, p->M);
psiCGMY(p, aux, &aux1);
aux2=Cmul(aux, ix);
aux1=CRmul(aux1,-p->T);
aux1=Cadd(aux1,aux2);
/////////////////end////////////////////////
/////////fraction computation/////////////////
aux2 =Cdiv(Cexp(aux1),aux);
aux.i=aux.i+1.;
aux3 =Cdiv(aux2,aux);
/////////////////end////////////////////////
////////////chi derivative computation/////////////
aux.i=-eta/(p->M-1);
aux.r=(p->M+p->om)/(p->M-1);
aux1=Cexp(RCmul(p->alp-1,Clog(aux)));
/////////////////end//////////////////////////////
aux2=Cmul(aux3,aux1);
return aux2.r*(p->alp);
}

static double ReDPutInt(double eta,  void *pp) ///Re( iexp(ixChi_+(eta+i om)-T p
{

params_cgmy *p = (params_cgmy*)pp;

dcomplex ix, aux, aux1, aux2, aux3;

ix.i=p->x;
ix.r=0.;
//////////////exponent computation////////////////////////
aux=chip(eta,p->alp, p->om, p->G);
psiCGMY(p, aux, &aux1);
aux2=Cmul(aux, ix);
aux1=CRmul(aux1,-p->T);
aux1=Cadd(aux1,aux2);
/////////////////end////////////////////////
/////////fraction computation/////////////////
aux2 =Cexp(aux1);
aux.i=aux.i+1.;
aux3 =Cdiv(aux2,aux);
```

```
/////////////////end//////////////////////////
////////////chi derivative computation/////////////
aux.i=eta/(p->G);
aux.r=1-(p->om)/(p->G);
aux1=Cexp(RCmul(p->alp-1,Clog(aux)));
/////////////////end//////////////////////////////
aux2=Cmul(aux3,aux1);
return -aux2.i*(p->alp);
}

static double ReDCallInt(double eta,  void *pp) ///Re( i exp(ixChi_-(eta+i om)-T
{

params_cgmy *p = (params_cgmy*)pp;

dcomplex ix, aux, aux1, aux2, aux3;

ix.i=p->x;
ix.r=0.;
//////////////exponent computation//////////////////////
aux=chim(eta,p->alp, p->om, p->M);
psiCGMY(p, aux, &aux1);
aux2=Cmul(aux, ix);
aux1=CRmul(aux1,-p->T);
aux1=Cadd(aux1,aux2);
/////////////////end//////////////////////////
/////////fraction computation/////////////////
aux2 =Cexp(aux1);
aux.i=aux.i+1.;
aux3 =Cdiv(aux2,aux);
/////////////////end//////////////////////////
////////////chi derivative computation/////////////
aux.i=-eta/(p->M-1);
aux.r=(p->M+p->om)/(p->M-1);
aux1=Cexp(RCmul(p->alp-1,Clog(aux)));
/////////////////end//////////////////////////////
aux2=Cmul(aux3,aux1);
return -aux2.i*(p->alp);
}

static int ParFt(double Spot, double Strike, double T, double r, double divid,
```

```
                  double C, double G, double M, double Y, int
                  iscall,long N, double *price,double *delta)
{
params_cgmy p;
double eps;
double UP_PARFT;
PnlFunc  Put, Call;

eps = 0.00001;
Set_params(&p, r, divid, Y, M, G, C, T, Strike, Spot, eps, iscall, N);
if (iscall)//Call
    {
if (p.x>0)
{
  iscall=0;
 Set_params(&p, r, divid, Y, M, G, C, T, Strike, Spot, eps, iscall, N);
 UP_PARFT = p.N*p.zet;
 Put.F =  &RePutInt;
 Put.params = &p;
 *price = pnl_integration (&Put, 0, UP_PARFT, p.N , "trap");
 *price =-*price*p.K/M_PI+(exp((r-divid)*T)*p.S - p.K);
         Put.F =  &ReDPutInt;
 Put.params = &p;
 *delta = pnl_integration (&Put, 0, UP_PARFT, p.N , "trap");
 *delta =exp(r*T)-*delta*p.K/(M_PI*p.S);
}
else {
   Set_params(&p, r, divid, Y, M, G, C, T, Strike, Spot, eps, iscall, N);
 UP_PARFT = p.N*p.zet;
 Call.F =  &ReCallInt;
 Call.params = &p;
 *price = pnl_integration (&Call, 0.0, UP_PARFT, p.N , "trap");
 *price =-*price*p.K/M_PI;
 Call.F =  &ReDCallInt;
 Call.params = &p;
 *delta = pnl_integration (&Call, 0.0, UP_PARFT, p.N , "trap");
 *delta =-*delta*p.K/(M_PI*p.S);
}
}
else   // Put
 {  if (p.x<0.)
```

```
  { iscall=0;
 Set_params(&p, r, divid, Y, M, G, C, T, Strike, Spot, eps, iscall, N);
        UP_PARFT = p.N*p.zet;
 Call.F =  &ReCallInt;
 Call.params = &p;
 *price = pnl_integration (&Call, 0.0, UP_PARFT, p.N , "trap");
 *price =-*price*p.K/M_PI-(exp((r-divid)*T)*p.S - p.K);
 Call.F =  &ReDCallInt;
 Call.params = &p;
 *delta = pnl_integration (&Call, 0.0, UP_PARFT, p.N , "trap");
 *delta =-exp(r*T)-*delta*p.K/(M_PI*p.S);}
  else
  { Set_params(&p, r, divid, Y, M, G, C, T, Strike, Spot, eps, iscall, N);
        UP_PARFT = p.N*p.zet;
 Put.F =  &RePutInt;
 Put.params = &p;
 *price = pnl_integration (&Put, 0, UP_PARFT, p.N , "trap");
 *price =-*price*p.K/M_PI;
        Put.F =  &ReDPutInt;
 Put.params = &p;
 *delta = pnl_integration (&Put, 0, UP_PARFT, p.N , "trap");
 *delta =-*delta*p.K/(M_PI*p.S);}
}
*price = exp(-r*T)**price;
*delta = exp(-r*T)**delta;

  return OK;
}

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

  iscall = FALSE;
  if (ptOpt->PayOff.Val.V_NUMFUNC_1->Compute == &Call) iscall = TRUE;

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

```
  return ParFt(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->C.Val.V_PDOUBLE,
               ptMod->G.Val.V_PDOUBLE,
               ptMod->M.Val.V_PDOUBLE,
               ptMod->Y.Val.V_PDOUBLE,
               iscall,
 Met->Par[0].Val.V_PINT,
               &(Met->Res[0].Val.V_DOUBLE),&(Met->Res[1].Val.V_DOUBLE));
}

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

  return  WRONG;
}

#endif

static int MET(Init)(PricingMethod *Met, Option *Opt)
{
  if (Met->init == 0)
    {
      Met->Par[0].Val.V_PINT = 300;
      Met->init = 1;
      Met->HelpFilenameHint = "ap_parft_cgmy1d_euro";

    }
  return OK;
}


PricingMethod MET(AP_PARFT) =
{
  "AP_PARFT",
  { {"N", LONG, {100}, ALLOW},
```

```
    {" ", PREMIA_NULLTYPE, {0}, FORBID}},
  CALC(AP_PARFT),
  { {"Price", DOUBLE, {100}, FORBID},
 {"Delta", DOUBLE, {100}, FORBID},
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CHK_OPT(AP_PARFT),
  CHK_ok,
  MET(Init)
};
```