Help

```
#include "
href../../../mod/cgmy1d/cgmy1d_lim/cgmy1d_lim_h_src.pdfcgmy1d_lim.h"
#include <pnl/pnl_mathtools.h>
#include <pnl/pnl_complex.h>
#include"pnl/pnl_specfun.h"
#include "pnl/pnl_integration.h"
#include <pnl/pnl_vector.h>
#include <pnl/pnl_matrix.h>
#include <pnl/pnl_fft.h>

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2019+2) //The "#els

static int CHK_OPT(AP_BPROJ)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(AP_BPROJ)(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 RNmu;
double T;
double q;
double c1;
double c2;
double c4;
double RePart;
double om;

} params_cgmy;
```

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

p->om = 0;
if ((call == 1) && (down == 1)){ p->om = MM > 2. ? 2. : (MM + 1.) / 2.; }
p->C = C;    //in the TSL parametrization C=C, Y=alphaplus=alphaminus, G=lambdam
p->G = G + p->om;
p->M = MM - p->om;
p->Y = Y;
p->T = t;
p->q = divid;
p->RNmu = r - divid + pnl_sf_gamma(-Y)*C*(exp(Y*log(MM)) - exp(Y*log(MM - 1))) +
p->c1 = p->RNmu + C*pnl_sf_gamma(1 - Y)*(exp((Y - 1)*log(MM)) - exp((Y - 1)*log(
p->c2 = C *pnl_sf_gamma(2 - Y)*(exp((Y - 2)*log(p->M)) + exp((Y - 2)*log(p->G)))
p->c4 = C *pnl_sf_gamma(4 - Y)*(exp((Y - 4)*log(p->M)) + exp((Y - 4)*log(p->G)))
p->RePart = pnl_sf_gamma(-Y)*C*(exp(Y*log(MM)) + exp(Y*log(G))) - p->RNmu*p->om;
}

static void SYMB_RN_CGMY(params_cgmy *p, double u, dcomplex *res)//computation f
{
dcomplex aux1, aux2, aux3, aux4;

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

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

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

res->r = p->RePart - p->C*pnl_sf_gamma(-p->Y)*(aux3.r + aux4.r);
res->i = -p->C*pnl_sf_gamma(-p->Y)*(aux3.i + aux4.i) + p->RNmu*u;
}


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

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

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

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

aux1.r = -p->RePart*p->T + p->C*p->T*pnl_sf_gamma(-p->Y)*(aux3.r + aux4.r);
aux1.i = p->C*p->T*pnl_sf_gamma(-p->Y)*(aux3.i + aux4.i) + p->T*p->RNmu*u;
aux2 = Cexp(aux1);
res->r = aux2.r;
res->i = aux2.i;
}


static double getTruncationAlpha(double T, double L1, void *pp)
{
params_cgmy *p = (params_cgmy*)pp;
return L1*sqrt(ABS(p->c2*T) + sqrt(ABS(p->c4*T)));

}


static int BPROJ_alpha(int UseCumulant, int call, int down, double S_0, double W
double C, double G, double M, double Y, double T, double rebate, double *price,

{
// call = 1 for call;
// down = 1 for down and out
// H = barrier

params_cgmy p;

int logN, P, Pbar;
double alph;// alph = size of valuation grid(width of projected density support
long int N;  // N = number of basis elements
int mult = 1;  // How much overhang to remove aliasing(hardcoded to one for sing
int interp_Atend = 0;  //see grid determination : this may get set to 1 below, i
```

```
long int K, nnot, i, m;
double dx, dt, nrdt, nqdt, a, l, xmin, dd;
double q_minus, q_plus, b3, b4; //  Gaussian Quad Constants
double sigma, sigma_plus, sigma_minus, es1, es2, dbar_0, dbar_1, d_0, d_1; /// a
double lws, rho, zeta, a2, zmin, aux, grand0;
double zeta_plus, zeta_minus, rho_plus, rho_minus;
double ed1, ed2, ed3;
long int nbar, Nmult;
double Cons, dw;
double varthet_01, varthet_m10, varthet_star, varthet_01R, varthet_starW;///// P
dcomplex chf;

PnlVect *Thet, *beta, *val_rebate, *Val, *Thetbar1, *Thetbar2;
PnlVectComplex *grand, *toepM, *toepR, *toepL;



////////////////////////////////
// APPROACH 1: Cumulant Based approach for grid width
// (see "Robust Option Pricing with Characteritics Functions and the BSpline Ord
////////////////////////////
if (UseCumulant == 1) // With cumulant based rule, choose N and Alpha(N = 2 ^ (P
{
logN = 14;  //14      //Uses N = 2 ^ logN  gridpoint
//L1 = 12;          // determines grid witdth(usually set L1 = 8 to 15 for Levy,
}
////////////////////////////////////
//// APPROACH 2: Manual GridWidth approach
////////////////////////////////////
else                  //Manually specify resolution and Pbar
{
P = 7;           // resolution is 2 ^ P
Pbar = 3;        // Determines density truncation grid with, 2 ^ Pbar
}


///// Algorithm parameters setup
////////////////////////////////////
Set_params(&p, r, q, Y, M, G, C, T / M0, call, down);

if (UseCumulant == 1) // Choose density truncation width based on cumulants
```

```
{
alph = getTruncationAlpha(T, L1, &p);
}
else    // Manually supply density truncation width above
{
logN = P + Pbar;
alph = pow(2, Pbar) / 2;
}
N = (long int)pow(2, logN);    // grid roughly centered on[c1 - alph, c1 + alph]
////////////////////////////////////////////////////////////////////

K = N / 2;

dx = 2 * alph / (N - 1); a = 1 / dx;

dt = T / M0;
nrdt = -r*dt;  nqdt = -q*dt;

Thet = pnl_vect_create_from_zero(2 * K); ///// space points
Thetbar1 = pnl_vect_create_from_zero(2 * K);
Thetbar2 = pnl_vect_create_from_zero(2 * K);
val_rebate = pnl_vect_create_from_zero(K); ///// Value part from rebate
Val = pnl_vect_create_from_zero(K); ///// Value function

Nmult = mult*N;
beta = pnl_vect_create_from_zero(Nmult);
grand = pnl_vect_complex_create_from_zero(Nmult);
toepM = pnl_vect_complex_create_from_zero(Nmult);
toepR = pnl_vect_complex_create_from_zero(Nmult);
toepL = pnl_vect_complex_create_from_zero(Nmult);

//////  Gaussian Quad Constants
q_plus = (1. + sqrt(3. / 5.)) / 2;  q_minus = (1. - sqrt(3. / 5.)) / 2.;
b3 = sqrt(15); b4 = b3 / 10.;

///////////////////////////////////
/////// DOWN & OUT
///////////////////////////////////
if (down == 1)
{
l = log(H / S_0);
```

```
xmin = l;
nnot = (long int)floor(1 - xmin*a);

/* if (nnot >= K) */
/*  printf("nnot is %.0 while K is %.0f, need to increase alpha \ n", nnot, K);

if ((call == 1) && (nnot == 1)) // In this case a DOC with H near S_0 is still v
{
interp_Atend = 1; //Instruct to use interpolation at algorithm end
//no change is made to dx
}
else
{
nnot = MAX(2, (long int)floor(1 - xmin*a));
dx = l / (1. - nnot);
}

a = 1. / dx;

lws = log(W / S_0);
nbar = (long int)floor(a*(lws - xmin) + 1);
rho = lws - (xmin + (nbar - 1)*dx);
zeta = a*rho;

a2 = pow(a, 2);
zmin = (1 - K)*dx;  //Kbar corresponds to zero

// Extend Pbar, only to invert(aliasing)
Cons = 24 * a2*exp(nrdt) / (double)Nmult;
dw = 2 * M_PI*a / (double)Nmult;

grand0 = 0.;

cf_RN_CGMY(&p, grand0, &chf);
LET_COMPLEX(grand, 0) = CRmul(chf, 1 / (24 * a2));
for (i = 1; i < Nmult; i++)
{
grand0 = grand0 + dw;
aux = pow(sin(grand0 / (2 * a)) / grand0, 2) / (2 + cos(grand0 / a));
cf_RN_CGMY(&p, grand0, &chf);
LET_COMPLEX(grand, i) = Cmul(Cexp(CRmul(CI, -zmin*grand0)), CRmul(chf, aux));
```

```
}
pnl_fft_inplace(grand);
for (i = 0; i < K; i++)
{
LET(beta, i) = Cons*pnl_vect_complex_get_real(grand, K - 1 - i);
}
LET(beta, K) = 0;
for (i = K + 1; i < 2 * K; i++)
{
LET(beta, i) = Cons*pnl_vect_complex_get_real(grand, 3 * K - i - 1);
}

pnl_real_fft(beta, toepM);
///// PAYOFF CONSTANTS---------------------------------- -
dx = (1 - p.om) / a;
varthet_01 = exp(.5*dx)*(5 * cosh(b4*dx) - b3*sinh(b4*dx) + 4) / 18;
varthet_m10 = exp(-.5*dx)*(5 * cosh(b4*dx) + b3*sinh(b4*dx) + 4) / 18;
varthet_star = varthet_01 + varthet_m10;
dx = 1. / a;
///-------------------------------------------------- -

////////////////////////////////////
//////// DOC
////////////////////////////////////
if (call == 1)
{
dx = (1 - p.om) / a;

sigma = 1 - zeta; sigma_plus = (q_plus - .5)*sigma; sigma_minus = (q_minus - .5)

es1 = exp(dx*sigma_plus); es2 = exp(dx*sigma_minus);

d_0 = exp(((1 - p.om)*rho + dx)*.5)*pow(sigma, 2) / 18 * (5 * ((1 - q_minus)*es2
d_1 = exp(((1 - p.om)*rho - dx)*.5)*sigma / 18 * (5 * ((.5*(zeta + 1) + sigma_mi

////////computation of dbar_0 and dbar_1 when the exponential weight is taken in
dx = (-p.om) / a;

for (i = K - 1; i >= 0; i--) LET(val_rebate, i) = exp(-p.om*xmin + dx*i); // NOT

es1 = exp(dx*sigma_plus); es2 = exp(dx*sigma_minus);
```

```
dbar_0 = exp(((-p.om)*rho + dx)*.5)*pow(sigma, 2) / 18 * (5 * ((1 - q_minus)*es2
dbar_1 = exp(((-p.om)*rho - dx)*.5)*sigma / 18 * (5 * ((.5*(zeta + 1) + sigma_mi

varthet_01R = exp(.5*dx)*(5 * cosh(b4*dx) - b3*sinh(b4*dx) + 4) / 18;
varthet_m10 = exp(-.5*dx)*(5 * cosh(b4*dx) + b3*sinh(b4*dx) + 4) / 18;
varthet_starW = varthet_01R + varthet_m10;

dx = 1 / a;

if (rebate > 0)
{
for (i = 1; i < nbar - 1; i++) LET(Thet, i) = -GET(val_rebate, i)*rebate*varthet
}

LET(Thet, nbar - 1) = exp(-p.om*(xmin + dx*(nbar - 1)))*(W*(exp(-rho)*d_0 - dbar
LET(Thet, nbar) = exp(-p.om*(xmin + dx*nbar))*(W*(exp(dx - rho)*(varthet_01 + d_

for (i = nbar + 1; i < K; i++) LET(Thet, i) = exp(-(p.om - 1)*(xmin + dx*i))*S_0

LET(Thet, 0) = -0.5*GET(val_rebate, 0)*rebate*varthet_starW;

////
LET_COMPLEX(toepR, 0) = Complex(Cons*pnl_vect_complex_get_real(grand, 2 * K - 1)
for (i = 1; i < K; i++)
{
LET_COMPLEX(toepR, i) = Complex(GET(beta, K + i), 0.);
}

pnl_fft_inplace(toepR);

/// computation of the sum_{k=K+1}^{k=K+n}beta_{k+K-n}theta_k
for (i = 0; i < K; i++)
{
LET(Thetbar1, i) = exp(-p.om*(xmin + dx*(K + i)));
LET(Thetbar2, i) = exp((1 - p.om)*(xmin + dx*(K + i)));
}
pnl_vect_mult_scalar(Thetbar1, (W + rebate)*varthet_starW);
pnl_vect_mult_scalar(Thetbar2, S_0*varthet_star);
```

```
pnl_real_fft(Thetbar1, grand);
for (i = 0; i < 2 * K; i++) LET_COMPLEX(grand, i) = Cmul(GET_COMPLEX(toepR, i),
pnl_ifft_inplace(grand);
for (i = 0; i < K; i++) LET(Thetbar1, i) = pnl_vect_complex_get_real(grand, i);

pnl_real_fft(Thetbar2, grand);
for (i = 0; i < 2 * K; i++) LET_COMPLEX(grand, i) = Cmul(GET_COMPLEX(toepR, i),
pnl_ifft_inplace(grand);
for (i = 0; i < K; i++) LET(Thetbar2, i) = pnl_vect_complex_get_real(grand, i);

pnl_real_fft(Thet, grand);
for (i = 0; i < 2 * K; i++) LET_COMPLEX(grand, i) = Cmul(GET_COMPLEX(toepM, i),
pnl_ifft_inplace(grand);


for (i = 0; i < K; i++)
{
LET(Val, i) = pnl_vect_complex_get_real(grand, i) + (GET(Thetbar2, i) - GET(Thet
}

if (rebate > 0)
{
for (i = 0; i < K; i++)
{
LET(Val, i) = GET(Val, i) - rebate*(1 - exp(nrdt))*GET(val_rebate, i);
}

}


for (m = M0 - 2; m >= 0; m--)
{
for (i = 1; i < K - 1; i++)
{
LET(Thet, i) = (GET(Val, i - 1) + 10 * GET(Val, i) + GET(Val, i + 1)) / 12.;
}
LET(Thet, 0) = (13 * GET(Val, 0) + 15 * GET(Val, 1) - 5 * GET(Val, 2) + GET(Val,
LET(Thet, K - 1) = (13 * GET(Val, K - 1) + 15 * GET(Val, K - 2) - 5 * GET(Val, K

LET(Thet, 0) = GET(Thet, 0) - 0.5*GET(val_rebate, 0)*rebate*varthet_starW;   //
```

```
pnl_real_fft(Thet, grand);
for (i = 0; i < 2 * K; i++) LET_COMPLEX(grand, i) = Cmul(GET_COMPLEX(toepM, i),
pnl_ifft_inplace(grand);

for (i = 0; i < K; i++)
{
LET(Val, i) = pnl_vect_complex_get_real(grand, i) + exp(nqdt*(M0 - m - 1))*(GET(

}

if (rebate > 0)
{
for (i = 0; i < K; i++)
{
LET(Val, i) = GET(Val, i) - rebate*(1 - exp(nrdt))*GET(val_rebate, i);
}
}
}
pnl_vect_div_vect_term(Val, val_rebate);
pnl_vect_plus_scalar(Val, rebate);
}//end DOC
else // put
{
if (rebate > 0)
{
LET(val_rebate, K - 1) = 0;
for (i = K - 2; i >= 0; i--) LET(val_rebate, i) = GET(beta, i + 1) + GET(val_reb
}
pnl_vect_mult_scalar(val_rebate, rebate);

zeta_plus = zeta*q_plus; zeta_minus = zeta*q_minus;
rho_plus = rho*q_plus; rho_minus = rho*q_minus;

ed1 = exp(rho_minus); ed2 = exp(rho / 2); ed3 = exp(rho_plus);

dbar_1 = zeta*zeta/ 2;
dbar_0 = zeta - dbar_1;        //  dbar_1 = zeta + .5*((zeta - 1) ^ 2 - 1);
d_0 = zeta*(5 * ((1 - zeta_minus)*ed1 + (1 - zeta_plus)*ed3) + 4 * (2 - zeta)*ed
d_1 = zeta*(5 * (zeta_minus*ed1 + zeta_plus*ed3) + 4 * zeta*ed2) / 18.;
```

```
LET(Thet, 0) = W / 2 - H*varthet_01;
LET(Thet, nbar - 1) = W*(.5 + dbar_0 - exp(-rho)*(varthet_m10 + d_0));
LET(Thet, nbar) = W*(dbar_1 - exp(-rho)*d_1);

for (i = 1; i < nbar - 1; i++) LET(Thet, i) = W - exp(xmin + dx*i)*S_0*varthet_s

LET(Thet, 0) = GET(Thet, 0) + 0.5*rebate;

//////////

pnl_real_fft(Thet, grand);
for (i = 0; i < 2 * K; i++) LET_COMPLEX(grand, i) = Cmul(GET_COMPLEX(toepM, i),
pnl_ifft_inplace(grand);

if (rebate > 0)
{
for (i = 0; i < K; i++)
{
LET(Val, i) = pnl_vect_complex_get_real(grand, i) + GET(val_rebate, i);
}
}
else
{
for (i = 0; i < K; i++)
{
LET(Val, i) = pnl_vect_complex_get_real(grand, i);
}
}

for (m = M0 - 2; m >= 0; m--)
{
for (i = 1; i < K - 1; i++)
{
LET(Thet, i) = (GET(Val, i - 1) + 10 * GET(Val, i) + GET(Val, i + 1)) / 12.;
}
LET(Thet, 0) = (13 * GET(Val, 0) + 15 * GET(Val, 1) - 5 * GET(Val, 2) + GET(Val,
LET(Thet, K - 1) = (13 * GET(Val, K - 1) + 15 * GET(Val, K - 2) - 5 * GET(Val, K

LET(Thet, 0) = GET(Thet, 0) + 0.5*rebate;   // account for overhang into the kno

pnl_real_fft(Thet, grand);
```

```
for (i = 0; i < 2 * K; i++) LET_COMPLEX(grand, i) = Cmul(GET_COMPLEX(toepM, i),
pnl_ifft_inplace(grand);

for (i = 0; i < K; i++)
{
LET(Val, i) = pnl_vect_complex_get_real(grand, i);
}

if (rebate > 0)
{
pnl_vect_plus_vect(Val, val_rebate);
}
}

}

}//end down
////////////////////////////////
////////////// UP & OUT
////////////////////////////////
else
{
double u;
lws = log(W / S_0);
u = log(H / S_0);
//nnot = min(K - 1, floor(K - a*u)); ///The min will always hold
nnot = (long int)floor(K - a*u);
if (!(call == 1) && (nnot == K - 1))
{
interp_Atend = 1; //dont change value of dx
}
else
{
dx = u / (K - nnot);  a = 1 / dx;
}


xmin = u - (K - 1)*dx;  /////NOTE: this used to be  xmin = u - (K - 2)*dx;
nbar = (long int)floor(a*(lws - xmin) + 1);

rho = lws - (xmin + (nbar - 1)*dx);
```

```
zeta = a*rho;

a2 = a*a;
zmin = (1 - K)*dx;  //Kbar corresponds to zero

/// Extend Pbar, only to invert(aliasing)
Cons = 24 * a2*exp(nrdt) / (double)Nmult;
dw = 2 * M_PI*a / (double)Nmult;

grand0 = 0.;

LET_COMPLEX(grand, 0) = Complex(1/(24 * a2), 0.);
for (i = 1; i < Nmult; i++)
{
grand0 = grand0 + dw;
aux = pow(sin(grand0 / (2 * a)) / grand0, 2) / (2 + cos(grand0 / a));
cf_RN_CGMY(&p, grand0, &chf);
LET_COMPLEX(grand, i) = Cmul(Cexp(CRmul(CI, -zmin*grand0)), CRmul(chf, aux));
}
pnl_fft_inplace(grand);
for (i = 0; i < K; i++)
{
LET(beta, i) = Cons*pnl_vect_complex_get_real(grand, K - 1 - i);
}
LET(beta, K) = 0;
for (i = K + 1; i < 2 * K; i++)
{
LET(beta, i) = Cons*pnl_vect_complex_get_real(grand, 3 * K - i - 1);
}

pnl_real_fft(beta, toepM);

///// PAYOFF CONSTANTS---------------------------------- -
varthet_01 = exp(.5*dx)*(5 * cosh(b4*dx) - b3*sinh(b4*dx) + 4) / 18;
varthet_m10 = exp(-.5*dx)*(5 * cosh(b4*dx) + b3*sinh(b4*dx) + 4) / 18;
varthet_star = varthet_01 + varthet_m10;
///------------------------------------------------ -

if (rebate > 0)
{
LET(val_rebate, 0) = Cons*pnl_vect_complex_get_real(grand, 2 * K - 1);
```

```
for (i = 1; i <K; i++) LET(val_rebate, i) = GET(beta, K+i) + GET(val_rebate, i -
}
pnl_vect_mult_scalar(val_rebate, rebate);
////////////////////////////////////
///// UOC
////////////////////////////////////
if (call == 1)
{
sigma = 1 - zeta; sigma_plus = (q_plus - .5)*sigma; sigma_minus = (q_minus - .5)
es1 = exp(dx*sigma_plus); es2 = exp(dx*sigma_minus);

dbar_0 = .5 + zeta*(.5*zeta - 1);
dbar_1 = sigma*(1 - .5*sigma);

d_0 = exp((rho + dx)*.5)*sigma*sigma / 18 * (5 * ((1 - q_minus)*es2 + (1 - q_plu
d_1 = exp((rho - dx)*.5)*sigma / 18 * (5 * ((.5*(zeta + 1) + sigma_minus)*es2 +

LET(Thet, K - 1) = H*varthet_m10 - .5*W;
LET(Thet, nbar - 1) = W*(exp(-rho)*d_0 - dbar_0);
LET(Thet, nbar) = W*(exp(dx - rho)*(varthet_01 + d_1) - (.5 + dbar_1));

for (i = nbar + 1; i < K - 1; i++) LET(Thet, i) = exp(xmin + dx*i)*S_0*varthet_s

LET(Thet, K - 1) = GET(Thet, K - 1) + 0.5*rebate;// account for overlap of right

////////////

pnl_real_fft(Thet, grand);
for (i = 0; i < 2 * K; i++) LET_COMPLEX(grand, i) = Cmul(GET_COMPLEX(toepM, i),
pnl_ifft_inplace(grand);

if (rebate > 0)
{
for (i = 0; i < K; i++)
{
LET(Val, i) = pnl_vect_complex_get_real(grand, i) + GET(val_rebate, i);
}
}
else
{
for (i = 0; i < K; i++)
```

```
{
LET(Val, i) = pnl_vect_complex_get_real(grand, i);
}
}

//////////////////////
for (m = M0 - 2; m >= 0; m--)
{
for (i = 1; i < K - 1; i++)
{
LET(Thet, i) = (GET(Val, i - 1) + 10 * GET(Val, i) + GET(Val, i + 1)) / 12.;
}
LET(Thet, 0) = (13 * GET(Val, 0) + 15 * GET(Val, 1) - 5 * GET(Val, 2) + GET(Val,
LET(Thet, K - 1) = (13 * GET(Val, K - 1) + 15 * GET(Val, K - 2) - 5 * GET(Val, K

LET(Thet, K-1) = GET(Thet, K-1) + 0.5*rebate;   // account for overhang into the

pnl_real_fft(Thet, grand);
for (i = 0; i < 2 * K; i++) LET_COMPLEX(grand, i) = Cmul(GET_COMPLEX(toepM, i),
pnl_ifft_inplace(grand);

for (i = 0; i < K; i++)
{
LET(Val, i) = pnl_vect_complex_get_real(grand, i);
}

if (rebate > 0)
{
pnl_vect_plus_vect(Val, val_rebate);
}
}
}
////////////////////////////////////
/////////// UOP
////////////////////////////////////
else ////(UOP)
{
zeta_plus = zeta*q_plus; zeta_minus = zeta*q_minus;
rho_plus = rho*q_plus; rho_minus = rho*q_minus;
ed1 = exp(rho_minus); ed2 = exp(rho / 2); ed3 = exp(rho_plus);
```

```
dbar_1 = zeta*zeta/ 2;
dbar_0 = zeta - dbar_1;        //  dbar_1 = zeta + .5*((zeta - 1) ^ 2 - 1);
d_0 = zeta*(5 * ((1 - zeta_minus)*ed1 + (1 - zeta_plus)*ed3) + 4 * (2 - zeta)*ed
d_1 = zeta*(5 * (zeta_minus*ed1 + zeta_plus*ed3) + 4 * zeta*ed2) / 18;

LET(Thet, nbar - 1) = W*(.5 + dbar_0 - exp(-rho)*(varthet_m10 + d_0));
LET(Thet, nbar) = W*(dbar_1 - exp(-rho)*d_1);

for (i = 0; i < nbar - 1; i++) LET(Thet, i) = W-exp(xmin + dx*i)*S_0*varthet_sta


LET(Thetbar1, K-1) = 0;
LET(Thetbar2, K-1) = exp(xmin - dx);

for (i = K+1; i < 2*K; i++)
{
LET_COMPLEX(toepL, i) = Complex(GET(beta, i-K), 0.);
}

pnl_fft_inplace(toepL);

for (i = K-2; i >=0; i--)
{
LET(Thetbar1, i) = GET(beta, i+1) + GET(Thetbar1, i + 1);
LET(Thetbar2, i) = exp(xmin - dx*(K-i));
}
pnl_vect_mult_scalar(Thetbar1, exp(r*dt)*W);
pnl_vect_mult_scalar(Thetbar2, exp(r*dt)*S_0*varthet_star);

pnl_real_fft(Thetbar2, grand);
for (i = 0; i < 2 * K; i++) LET_COMPLEX(grand, i) = Cmul(GET_COMPLEX(toepL, i),
pnl_ifft_inplace(grand);
for (i = 0; i < K; i++) LET(Thetbar2, i) = pnl_vect_complex_get_real(grand, i);

pnl_real_fft(Thet, grand);
for (i = 0; i < 2 * K; i++) LET_COMPLEX(grand, i) = Cmul(GET_COMPLEX(toepM, i),
pnl_ifft_inplace(grand);

if (rebate > 0)
{
for (i = 0; i < K; i++)
```

```
{
LET(Val, i) = pnl_vect_complex_get_real(grand, i) + exp(-r*dt)*(GET(Thetbar2, i)
}
}
else
{
for (i = 0; i < K; i++)
{
LET(Val, i) = pnl_vect_complex_get_real(grand, i) + exp(-r*dt)*(GET(Thetbar2, i)
}
}

for (m = M0 - 2; m >= 0; m--)
{
for (i = 1; i < K - 1; i++)
{
LET(Thet, i) = (GET(Val, i - 1) + 10 * GET(Val, i) + GET(Val, i + 1)) / 12.;
}
LET(Thet, 0) = (13 * GET(Val, 0) + 15 * GET(Val, 1) - 5 * GET(Val, 2) + GET(Val,
LET(Thet, K - 1) = (13 * GET(Val, K - 1) + 15 * GET(Val, K - 2) - 5 * GET(Val, K

LET(Thet, K-1) = GET(Thet, K-1) + 0.5*rebate;   // account for overhang into the

pnl_real_fft(Thet, grand);
for (i = 0; i < 2 * K; i++) LET_COMPLEX(grand, i) = Cmul(GET_COMPLEX(toepM, i),
pnl_ifft_inplace(grand);

for (i = 0; i < K; i++)
{
LET(Val, i) = pnl_vect_complex_get_real(grand, i) - exp(nqdt*(M0 - m - 1))*(GET(
}

if (rebate > 0)
{
pnl_vect_plus_vect(Val, val_rebate);
}
}
}
}//end up

if (interp_Atend == 1)
```

```
{
dd = 0 - (xmin + (nnot - 1)*dx);
*price = GET(Val, nnot-1) + (GET(Val, nnot) - GET(Val, nnot-1))*dd / dx; //ie li

}
else
{
*price = GET(Val, nnot-1);
}

*delta = (GET(Val, nnot-1) - GET(Val, nnot - 2)) / (S_0*exp(xmin + (nnot-1)*dx)-

pnl_vect_free(&Thet);
pnl_vect_free(&beta);
pnl_vect_free(&val_rebate);
pnl_vect_free(&Val);
pnl_vect_free(&Thetbar1);
pnl_vect_free(&Thetbar2);
pnl_vect_complex_free(&grand);
pnl_vect_complex_free(&toepM);
pnl_vect_complex_free(&toepL);
pnl_vect_complex_free(&toepR);

  return OK;
}

//=============================================================================
static int CALC(AP_BPROJ)(void *Opt,void *Mod,PricingMethod *Met)
{
TYPEOPT* ptOpt=( TYPEOPT*)Opt;
TYPEMOD* ptMod=( TYPEMOD*)Mod;
double r,divid,limit, strike, spot,rebate;

NumFunc_1 *p;
int down;
int ifCall;

r=log(1.+ptMod->R.Val.V_DOUBLE/100.);
divid=log(1.+ptMod->Divid.Val.V_DOUBLE/100.);
limit=((ptOpt->Limit.Val.V_NUMFUNC_1)->Compute)((ptOpt->Limit.Val.V_NUMFUNC_1)->
p=ptOpt->PayOff.Val.V_NUMFUNC_1;
```

```
strike=p->Par[0].Val.V_DOUBLE;
spot=ptMod->S0.Val.V_DOUBLE;
ifCall=((p->Compute)==&Call);

rebate=((ptOpt->Rebate.Val.V_NUMFUNC_1)->Compute)((ptOpt->Rebate.Val.V_NUMFUNC_1


if ((ptOpt->DownOrUp).Val.V_BOOL==DOWN)
down=1;
else down=0;

return BPROJ_alpha(1, ifCall, down,  spot, strike,
limit,Met->Par[0].Val.V_PINT, Met->Par[1].Val.V_PINT, r, divid, ptMod->C.Val.V_P
                ptMod->G.Val.V_PDOUBLE,
                ptMod->M.Val.V_PDOUBLE,
                ptMod->Y.Val.V_PDOUBLE,
ptOpt->Maturity.Val.V_DATE-ptMod->T.Val.V_DATE,  rebate,
&(Met->Res[0].Val.V_DOUBLE), &(Met->Res[1].Val.V_DOUBLE));

}

static int CHK_OPT(AP_BPROJ)(void *Opt, void *Mod)
{
    Option *ptOpt = (Option *)Opt;
  TYPEOPT *opt = (TYPEOPT *)(ptOpt->TypeOpt);

   if ((opt->OutOrIn).Val.V_BOOL == OUT)
    if ((opt->Parisian).Val.V_BOOL == FALSE)
      if ((opt->EuOrAm).Val.V_BOOL == EURO)
        return  OK;
   return WRONG;
}

#endif

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

  if (Met->init == 0)
    {
      Met->Par[0].Val.V_PINT = 252;
```

```
  Met->Par[1].Val.V_PINT = 12;
      Met->init = 1;
      Met->HelpFilenameHint = "ap_proj_cgmy1d";

    }
  return OK;
}


PricingMethod MET(AP_BPROJ) =
{
  "AP_KIRKBY_BPROJ",
  { {"Number of discrete monitoring points",INT,{100},ALLOW},
{"Scale of logprice range", DOUBLE, {100}, ALLOW},
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CALC(AP_BPROJ),
  { {"Price", DOUBLE, {100}, FORBID},
 {"Delta", DOUBLE, {100}, FORBID},
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CHK_OPT(AP_BPROJ),
  CHK_ok,
  MET(Init)
};
```