Help

```
#include "
href../../../common/math/kirkby/model_proj_h_src.pdfmodel_proj.h"
#include <pnl/pnl_mathtools.h>
#include "pnl/pnl_specfun.h"

double Model_proj::get_truncation_alpha(double L1, double T) const{
return L1 * sqrt(ABS(c2 * T) + sqrt(ABS(c4 * T)));
}

////////////////////
// CGMY Model
////////////////////

void CGMY_Model_proj::setParams(double C_, double G_, double M_, double Y_, doub
C = C_;
G = G_ + om_;
M = M_ - om_;
Y = Y_;

om = om_;

// TODO: double check this, he uses separate MM and G see below, ask Oleg
RNmu = r - q + pnl_sf_gamma(-Y) * C * (exp(Y * log(M_)) - exp(Y * log(M_ - 1)))
c1 = RNmu + C * pnl_sf_gamma(1 - Y) * (exp((Y - 1) * log(M_)) - exp((Y - 1) * lo
c2 = C * pnl_sf_gamma(2 - Y) * (exp((Y - 2) * log(M)) + exp((Y - 2) * log(G)));
c4 = C * pnl_sf_gamma(4 - Y) * (exp((Y - 4) * log(M)) + exp((Y - 4) * log(G)));

RePart = pnl_sf_gamma(-Y) * C * (exp(Y * log(M_)) + exp(Y * log(G_))) - RNmu * o
}

void CGMY_Model_proj::rn_chf(double t, double u, dcomplex* res) const
{
dcomplex aux1, aux2, aux3, aux4;

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

aux2.r = G;
aux2.i = u;
```

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

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


////////////////////
// Black-Scholes Model
////////////////////

void BlackScholes_Model_proj::rn_symbol(double u, dcomplex* res) const
{
res->r = -0.5 * sigma * sigma * u * u;
res->i = u * RNmu;
}

void BlackScholes_Model_proj::rn_chf(double t, double u, dcomplex* res) const
{
rn_symbol(u, res);
*res = Cexp(RCmul(t, *res));
}

void BlackScholes_Model_proj::bs_roots(double lam, double*bp, double*bm) const
{
double vv = 0.5*sigma*sigma;

*bp = (-RNmu + pow(RNmu*RNmu + 4 * lam*vv, 0.5)) / (2 * vv); ///lam=q+r
*bm = (-RNmu - pow(RNmu*RNmu + 4 * lam*vv, 0.5)) / (2 * vv);
}

void BlackScholes_Model_proj::setParams(double sigma_, double om_)
{
sigma = sigma_; om = om_;

RNmu = r - q - 0.5 * sigma * sigma;
```

```
c1 = RNmu;
c2 = sigma * sigma;
c4 = 0.0;

}
void BlackScholes_Model_proj::cf_phim_BS(double u, double bm, dcomplex *res) con
{
dcomplex aux1, aux2;

aux1.r = 0;
aux1.i = u;

aux2 = RCdiv(-bm, RCadd(-bm, aux1));

res->r = aux2.r;
res->i = aux2.i;
}


////////////////////
// Kou Model
////////////////////

void Kou_Model_proj::rn_symbol(double u, dcomplex* res) const
{
double sig2 = .5 * sigma * sigma;
double temp1 = r - sig2 - lam * ((1 - p_up) * eta2 / (eta2 + 1) + p_up * eta1 /

//temp2 = -sig2 * u * u + lam * ((1 - p_up) * eta2 / (eta2 + 1i * u) + p_up * et

// TODO: make this more efficient by not creating an new complex data point, ins
// NOTE: this is an inefficient implementation just to get it running

double t3 = -sig2 * u * u;

dcomplex t4 = RCdiv((1 - p_up) * eta2, Complex(eta2, u));
dcomplex t5 = RCdiv(p_up * eta1, Complex(eta1, -u));
dcomplex t6 = CRsub(Cadd(t4, t5), 1);
dcomplex temp2 = RCadd(t3, RCmul(lam, t6));

*res = Cadd(CRmul(CI, u * temp1), temp2);
```

```
}

void Kou_Model_proj::rn_chf(double t, double u, dcomplex* res) const
{
rn_symbol(u, res);
*res = Cexp(RCmul(t, *res));
}

void Kou_Model_proj::setParams(double sigma_, double lam_, double p_up_, double
{
sigma = sigma_; lam = lam_; p_up = p_up_; eta1 = eta1_; eta2 = eta2_; om = om_;

double sigma2 = sigma * sigma;

RNmu = r - q - 0.5 * sigma2 - lam * (p_up * eta1 / (eta1 - 1) + (1 - p_up) * eta
c1 = RNmu + lam * p_up / eta1 + lam * (1 - p_up) / eta2;
c2 = sigma2 + 2 * lam * p_up / (eta1 * eta1) + 2 * lam * (1 - p_up) / (eta2 * et
c4 = 24. * lam * (p_up / pow(eta1, 4) + (1 - p_up) / pow(eta2, 4));
}

////////////////////
// NIG Model
////////////////////

void NIG_Model_proj::rn_symbol(double u, dcomplex* res) const
{
dcomplex c1 = Csqrt(Complex(asq -bet*bet + u*u, -2*bet*u));
dcomplex yy = RCmul(-delt, CRsub(c1, sqd));
*res = Cadd(yy, Complex(0.0, u * RNmu));
}

void NIG_Model_proj::rn_chf(double t, double u, dcomplex* res) const
{
rn_symbol(u, res);
*res = Cexp(RCmul(t, *res));
}

void NIG_Model_proj::setParams(double alph_, double bet_, double delt_, double o
{
alph = alph_; bet = bet_; delt = delt_; om = om_;
```

```
asq = alph * alph;
bsq = bet  * bet;
sqd = sqrt(asq - bsq);

RNmu = r - q + delt * (sqrt(asq - pow(bet + 1, 2)) - sqd);
c1 = RNmu + delt * bet / sqd;
c2 = delt * asq * pow(asq - bsq, -1.5);
c4 = 3 * delt * asq * (asq + 4 * bsq) * pow(asq - bsq, -3.5);
}
```