Help

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <
href../../../common/math/cdo/cdo_math_h_src.pdfmath.h>

#include "
href../../../common/math/read_market_zc/InitialYieldCurve_h_src.pdfmath/read_mar
#include "
href../../../mod/sg1d/sg1d_stdi/QuadraticModel_h_src.pdfQuadraticModel.h"
#include "
href../../../common/optype_h_src.pdfoptype.h"
#include "
href../../../common/enums_h_src.pdfenums.h"


#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2007+2) //The "#els
#else

// Compute the initial rate r_0 and corresponding value x_0
void initial_short_rate(ZCMarketData *ZCMarket, double *r0, double *x0)
{
  if (ZCMarket->FlatOrMarket == 0) *r0 = ZCMarket->Rate;

  else *r0 = -log(BondPrice(INC, ZCMarket)) / INC;

  *x0 = sqrt(2.* (*r0));
}


// bond_coeffs computes P(0,T) coefficients and their derivatives
void bond_coeffs(ZCMarketData *ZCMarket, Data *data, double T,  double beta, dou
{
  int j, Nbr_Step_Integration;
  double gamma, h, db, dc, V, dB, f0_s, s, dt;

  gamma = sqrt(SQR(beta) + pow(sigma, 2));

  // Here, we compute V(0,T), B(0,T) and its derivatives
```

```
  h = 1 / ((beta + gamma) * exp(2 * gamma * T) + gamma - beta);
  data->T = T;
  data->B = (exp(2 * gamma * T) - 1) * h;
  data->dB = SQR(2 * gamma * exp(gamma * T) * h);
  data->V = SQR(sigma) * data->B;

  // Instantaneous forward rate f(0, T)
  f0_s = ForwardRate(T, ZCMarket);
  data->f0_T = f0_s;

  // Value of db and dc at time T
  db = -data->dB * x0 + sqrt(data->dB * (2 * f0_s - data->V));
  dc = 0.5 * (SQR(db) / data->dB + data->V);
  data->db = db;


  Nbr_Step_Integration = floor(T * 1000); // Number of steps in the trapezoidal
  dt = T / Nbr_Step_Integration;
  data->b = (0.5 * db) * dt;
  data->c = (0.5 * dc) * dt;

  // Integration of db and dc from 0 to T using trapezoidal rule.
  for (j = 1; j < Nbr_Step_Integration; j++)
    {
      s = j * dt;

      h = 1 / ((beta + gamma) * exp(2 * gamma * s) + gamma - beta);
      V = SQR(sigma) * (exp(2 * gamma * s) - 1) * h;
      dB = SQR(2 * gamma * exp(gamma * s) * h);

      f0_s = ForwardRate(s, ZCMarket);

      db = -dB * x0 + sqrt(dB * (2 * f0_s - V));
      dc = 0.5 * (SQR(db) / dB + V);

      data->b += db * dt;
      data->c += dc * dt;
    }

}
```

```
// Gives the omega distribution of the zero-coupon bond P(T, S) data1 contains t
void transport(Omega *om, Data data1, Data data2, double beta, double sigma, dou
{
  double a, c;

  om->B = (data2.B - data1.B) / (data1.dB - data1.V * (data2.B - data1.B));
  om->b = om->B * sqrt(data1.dB) * ((data2.b - data1.b) / (data2.B - data1.B) -
  a = data1.db / sqrt(data1.dB);
  c = 0.5 * (log(1 + data1.V * om->B) + (om->B * SQR(a) + 2 * om->b * a - data1.
  om->c = data2.c - data1.c - c;

  om->mu = sqrt(data1.dB) * x0 + data1.db / sqrt(data1.dB);
  om->V = data1.V;
}

// Transform Omega distribution to a chi^2 distribution
void om2chn(Omega om, Chn *chn)
{

  chn->nu = 1; // in the simple factor case, nu is always equal to 1
  chn->lambda = SQR(om.mu + om.b / om.B) / om.V;
  chn->beta = .5 * om.V * om.B;
  chn->alpha = om.c - .5 * SQR(om.b) / om.B;

}


#endif //PremiaCurrentVersion
```