Help

```
#include  "
href../../../mod/sg1d/sg1d_stdi/sg1d_stdi_h_src.pdfsg1d_stdi.h"
#include  "
href../../../mod/sg1d/sg1d_stdi/QuadraticModel_h_src.pdfQuadraticModel.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2007+2) //The "#els
static int CHK_OPT(MC_ZBO)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(MC_ZBO)(void *Opt, void *Mod, PricingMethod *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else
/*Call Option*/
static int mc_quad1d(double flat_flag, double alpha, double beta, double t, doub
{
  double x0, K, rr;
  double bond_price, g, price_sample;
  double mean_price, var_price;
  int i;
  int init_mc;
  int simulation_dim = 1;
  double alpha_int, z_alpha;

  Data data1, data2;
  Omega om;


  /* Value to construct the confidence interval */
  alpha_int = (1. - confidence) / 2.;
  z_alpha = pnl_inv_cdfnor(1. - alpha_int);

  /*MC sampling*/
  init_mc = pnl_rand_init(generator, simulation_dim, M);
  if (init_mc == OK)
    {
```

```
x0 = sqrt(2.*r0);
if (flat_flag == 0)
  x0 = sqrt(2.*r0);
else
  spot_rate(&rr, &x0);

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

bond_coeffs(&data1, T, flat_flag, alpha, beta, sigma, x0);
/* coefficients of P(0,T) */
bond_coeffs(&data2, S, flat_flag, alpha, beta, sigma, x0);
/* coefficients of P(0,s) */

/* omega distribution of P(s,T) under the T-forward measure */
transport(&om, data1, data2, alpha, beta, sigma, x0);

/* mean and variance of x(T) in the T-forward measure */
T_forward_distrib(&om, data1, x0);
mean_price = 0.0;
var_price = 0.;
for (i = 0; i < M; i++)
  {
    g = pnl_rand_normal(generator) * sqrt(om.V) + om.mu;
    bond_price = exp(-(.5 * om.B * SQR(g) + om.b * g + om.c));
    price_sample = data1.P * (Payoff->Compute)(Payoff->Par, bond_price);
    mean_price += price_sample / (double)M;
    var_price += SQR(price_sample) / (double)M;
  }

var_price -= SQR(mean_price);

/*Price*/
*ptprice = mean_price;

/*error*/
*pterror_price = var_price;

/* Price Confidence Interval */
*inf_price = *ptprice - z_alpha * (*pterror_price);
*sup_price = *ptprice + z_alpha * (*pterror_price);
```

```
      /*Delta*/
      /**ptdelta=0.;*/
    }


  return OK;
}


int CALC(MC_ZBO)(void *Opt, void *Mod, PricingMethod *Met)
{
  TYPEOPT *ptOpt = (TYPEOPT *)Opt;
  TYPEMOD *ptMod = (TYPEMOD *)Mod;

  return mc_quad1d(ptMod->flat_flag.Val.V_INT,
                   ptMod->alpha.Val.V_DOUBLE,
                   ptMod->beta.Val.V_DOUBLE,
                   ptMod->T.Val.V_DATE,
                   ptMod->Sigma.Val.V_PDOUBLE,
                   MOD(GetYield)(ptMod),
                   ptOpt->BMaturity.Val.V_DATE,
                   ptOpt->OMaturity.Val.V_DATE,
                   ptOpt->PayOff.Val.V_NUMFUNC_1,
                   Met->Par[0].Val.V_LONG,
                   Met->Par[1].Val.V_ENUM.value,
                   Met->Par[2].Val.V_PDOUBLE,
                   &(Met->Res[0].Val.V_DOUBLE),
                   &(Met->Res[1].Val.V_DOUBLE),
                   &(Met->Res[2].Val.V_DOUBLE),
                   &(Met->Res[3].Val.V_DOUBLE));
}

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

  return WRONG;
}

#endif //PremiaCurrentVersion
```

```
static int MET(Init)(PricingMethod *Met, Option *Opt)
{
  int type_generator;
  if (Met->init == 0)
    {
      Met->init = 1;

      Met->Par[0].Val.V_LONG = 10000;
      Met->Par[1].Val.V_ENUM.value = 0;
      Met->Par[1].Val.V_ENUM.members = &PremiaEnumMCRNGs;
      Met->Par[2].Val.V_DOUBLE = 0.95;


    }
  type_generator = Met->Par[1].Val.V_ENUM.value;


  if (pnl_rand_or_quasi(type_generator) == PNL_QMC)
    {
      Met->Res[2].Viter = IRRELEVANT;
      Met->Res[3].Viter = IRRELEVANT;
      Met->Res[4].Viter = IRRELEVANT;

    }
  else
    {
      Met->Res[2].Viter = ALLOW;
      Met->Res[3].Viter = ALLOW;
      Met->Res[4].Viter = ALLOW;
    }
  return OK;
}

PricingMethod MET(MC_ZBO) =
{
  "MC_Quadratic1d_ZBO",
  { {"N iterations", LONG, {100}, ALLOW},
    {"RandomGenerator(Quasi Random not allowed)", ENUM, {100}, ALLOW}, {"Confide
  },
  CALC(MC_ZBO),
  { {"Price", DOUBLE, {100}, FORBID}/*,{"Delta",DOUBLE,{100},FORBID}*/ , {"Error
```

```
    {"Sup Price", DOUBLE, {100}, FORBID} , {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CHK_OPT(MC_ZBO),
  CHK_mc,
  MET(Init)
} ;
```