```
   Help
#include "sg1d_stdi.h"
#include "math/read_market_zc/InitialYieldCurve.h"
#include "Quadraticmodel.h"


#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2007+2) //The "#else" part of the code will be freely av
    ailable after the (year of creation of this file + 2)
static int CHK_OPT(CF_ZCCallBondEuroSG1D)(void *Opt, void *
    Mod)
{
  return NONACTIVE;
}
int CALC(CF_ZCCallBondEuroSG1D)(void *Opt,void *Mod,Pricing
    Method *Met)
{
return AVAILABLE_IN_FULL_PREMIA;
}
#else

double zb_call_quad1d(ZCMarketData *ZCMarket, double beta,
    double sigma, double T, double S, double strike)
{
    double r0, x0, POT, POS;
    double call_price, p1, p2;

    Data data1, data2;
    Omega om;
    Chn chn;

    r0=0.0; x0=0.0; p1=0.0; p2=0.0;

    initial_short_rate(ZCMarket, &r0, &x0);

    // coefficients of P(0,T)
    bond_coeffs(ZCMarket, &data1, T, beta, sigma, x0);

    // coefficients of P(0,S)
    bond_coeffs(ZCMarket, &data2, S, beta, sigma, x0);
```

```
    // omega distribution of P(S,T)
    transport(&om, data1, data2, beta, sigma, x0);

    // transforms the omega distribution of P(s,T) into chi
    2 distribution
    om2chn(om, &chn);

    // Price of the Call option
    pnl_cdfbchi2n(-chn.alpha-log(strike), 1, chn.lambda/(1+
    2*chn.beta), chn.beta/(1+2*chn.beta), &p1);
   pnl_cdfbchi2n(-chn.alpha-log(strike), 1, chn.lambda, chn
    .beta, &p2);

    // Call Price
    P0T = BondPrice(T, ZCMarket);
    P0S = BondPrice(S, ZCMarket);
    call_price = P0S*p1 - P0T*strike*p2;

    if(call_price<0) call_price = 0.;

    return call_price;
}


static int zbc_quad1d(double flat_flag, double beta,
    double sigma, double r0, double S, double T, NumFunc_1 *p,
    double *price)
{
    double strike;
    ZCMarketData ZCMarket;

    if(flat_flag==0)
    {
        ZCMarket.FlatOrMarket = 0;
        ZCMarket.Rate = r0;
    }

    else
    {
        ZCMarket.FlatOrMarket = 1;
        ReadMarketData(&ZCMarket);
```

```
        if(T > GET(ZCMarket.tm,ZCMarket.Nvalue-1))
        {
            printf("{nError : time bigger than the last
    time value entered in initialyield.dat{n");
            exit(EXIT_FAILURE);
        }
    }


    strike = p->Par[0].Val.V_DOUBLE;

    *price = zb_call_quad1d(&ZCMarket, beta, sigma, T, S,
    strike);

    DeleteZCMarketData(&ZCMarket);

    return OK;
}


int CALC(CF_ZCCallBondEuroSG1D)(void *Opt,void *Mod,Pricing
    Method *Met)
{
  TYPEOPT* ptOpt=(TYPEOPT*)Opt;
  TYPEMOD* ptMod=(TYPEMOD*)Mod;

  return zbc_quad1d(    ptMod->flat_flag.Val.V_INT,
                        ptMod->a.Val.V_DOUBLE,
                        ptMod->Sigma.Val.V_PDOUBLE,
                        MOD(GetYield)(ptMod),
                        ptOpt->BMaturity.Val.V_DATE-ptMod->
    T.Val.V_DATE,
                        ptOpt->OMaturity.Val.V_DATE-ptMod->
    T.Val.V_DATE,
                        ptOpt->PayOff.Val.V_NUMFUNC_1,
                        &(Met->Res[0].Val.V_DOUBLE));
}

static int CHK_OPT(CF_ZCCallBondEuroSG1D)(void *Opt, void *
    Mod)
```

```
{
  return strcmp( ((Option*)Opt)->Name,"ZeroCouponCallBondEu
    ro");
}

#endif //PremiaCurrentVersion
static int MET(Init)(PricingMethod *Met,Option *Opt)
{
  if ( Met->init == 0)
    {
      Met->init=1;
       Met->HelpFilenameHint = "cf_quadratic1d_zbcalleuro";
    }

  return OK;
}

PricingMethod MET(CF_ZCCallBondEuroSG1D)=
{
  "CF_SquareGaussian1d_ZBCallEuro",
  {{" ",PREMIA_NULLTYPE,{0},FORBID}},
  CALC(CF_ZCCallBondEuroSG1D),
  {{"Price",DOUBLE,{100},FORBID},{" ",PREMIA_NULLTYPE,{0},
    FORBID}},
  CHK_OPT(CF_ZCCallBondEuroSG1D),
  CHK_ok,
  MET(Init)
} ;
```

References