```
   Help
#include <stdlib.h>
#include  "hullwhite1d_stdi.h"
#include "hullwhite1d_includes.h"

//The "#else" part of the code will be freely available aft
    er the (year of creation of this file + 2)
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2007+2)
int CALC(CF_ReceiverSwaptionHW1D)(void *Opt,void *Mod,Prici
    ngMethod *Met)
{
return AVAILABLE_IN_FULL_PREMIA;
}
static int CHK_OPT(CF_ReceiverSwaptionHW1D)(void *Opt, voi
    d *Mod)
{
  return NONACTIVE;
}
#else

///* Computation the function phi used to find the Criti
    cal Rate in the Jamishidian decomposition
static double phi(ZCMarketData* ZCMarket, double r, double
    periodicity, double option_maturity, double contract_matu
    rity, double SwaptionFixedRate, double a, double sigma)
{
    int i, nb_payement;
    double ci, sum,sum_der,ti;

    double ZCPrice;
    double A_tT, B_tT;

    ZCPrice = 0.;
    A_tT = 0; B_tT = 0;
    sum=0.;
    sum_der=0.;

    ci = periodicity * SwaptionFixedRate;
    ti = option_maturity;
    nb_payement = (int)((contract_maturity-option_maturity)
    /periodicity);

    for(i=1; i<=nb_payement; i++)
    {
        ti += periodicity;

        ZCPrice_CoefficientHW1D(ZCMarket, a, sigma, option_
    maturity, ti, &A_tT, &B_tT);

        ZCPrice = ZCPrice_Using_CoefficientHW1D(r, A_tT, B_
    tT);

        sum += ci * ZCPrice;

        sum_der += ci * ZCPrice * (-B_tT);
    }

    sum += ZCPrice;

    sum_der += ZCPrice * (-B_tT);

    return (sum-1.)/sum_der;
}


///* Computation of Critical Rate in the Jamishidian de
    composition, with the newton method to find zero of a function
static double Critical_Rate(ZCMarketData* ZCMarket, double
    r_initial, double periodicity, double option_maturity,
    double contract_maturity, double SwaptionFixedRate, double a,
    double sigma)
{
  double previous,current;
  int nbr_iterations;

  const double precision = 0.0001;

  current = r_initial;
  nbr_iterations = 0;
  do
    {
        nbr_iterations++;
        previous =current;
        current=current-phi(ZCMarket, current, periodicity,
     option_maturity, contract_maturity, SwaptionFixedRate, a,
     sigma);

    } while((fabs(previous-current) > precision) && (nbr_
    iterations <= 10));

  return current;
}

///* Payer Swaption price as a combination of ZC Call
    option prices
static int cf_ps1d(int flat_flag, double r_t, double Nomina
    l, double periodicity, double option_maturity, double contr
    act_maturity, double SwaptionFixedRate, double a, double si
    gma,double *price)
{
    int i, nb_payement;
    double ci, sum ,ti;

    double critical_r, Strike_i, CallOptionPrice;
    ZCMarketData ZCMarket;

    CallOptionPrice = 0.; /* to avoid warning */
    /* Flag to decide to read or not ZC bond datas in "ini
    tialyields.dat" */
    /* If P(0,T) not read then P(0,T)=exp(-r0*T) */
    if(flat_flag==0)
    {
        ZCMarket.FlatOrMarket = 0;
        ZCMarket.Rate = r_t;
    }

    else
    {
        ZCMarket.FlatOrMarket = 1;
        ReadMarketData(&ZCMarket);
        r_t = -log(BondPrice(INC, &ZCMarket))/INC;

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

    ti = option_maturity;
    ci = periodicity * SwaptionFixedRate;

    nb_payement = (int)((contract_maturity-option_maturity)
    /periodicity);

    critical_r = Critical_Rate(&ZCMarket, r_t, periodicity,
     option_maturity, contract_maturity, SwaptionFixedRate, a,
     sigma);

    sum=0.;

    for(i=1; i<=nb_payement; i++)
    {
        ti += periodicity;

        Strike_i = cf_hw1d_zcb(&ZCMarket, a, sigma, option_
    maturity, critical_r, ti);

        CallOptionPrice = cf_hw1d_zbcall(&ZCMarket, a, si
    gma, ti, option_maturity, Strike_i);

        sum += ci * CallOptionPrice;
    }

    sum += CallOptionPrice;

    *price = Nominal * sum;

    DeleteZCMarketData(&ZCMarket);
```

```
    return OK;
}

int CALC(CF_ReceiverSwaptionHW1D)(void *Opt,void *Mod,Prici
    ngMethod *Met)
{
    TYPEOPT* ptOpt=(TYPEOPT*)Opt;
    TYPEMOD* ptMod=(TYPEMOD*)Mod;


    return cf_ps1d( ptMod->flat_flag.Val.V_INT,
                    MOD(GetYield)(ptMod),
                    ptOpt->Nominal.Val.V_PDOUBLE,
                    ptOpt->ResetPeriod.Val.V_DATE,
                    ptOpt->OMaturity.Val.V_DATE-ptMod->T.
    Val.V_DATE,
                    ptOpt->BMaturity.Val.V_DATE-ptMod->T.
    Val.V_DATE,
                    ptOpt->FixedRate.Val.V_PDOUBLE,
                    ptMod->a.Val.V_DOUBLE,
                    ptMod->Sigma.Val.V_PDOUBLE,
                    &(Met->Res[0].Val.V_DOUBLE));
}
static int CHK_OPT(CF_ReceiverSwaptionHW1D)(void *Opt, voi
    d *Mod)
{
  return strcmp( ((Option*)Opt)->Name,"ReceiverSwaption");
}
#endif //PremiaCurrentVersion



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

  return OK;
```

```
}

PricingMethod MET(CF_ReceiverSwaptionHW1D)=
{
  "CF_HullWhite1d_ReceiverSwaption",
  {{" ",PREMIA_NULLTYPE,{0},FORBID}},
  CALC(CF_ReceiverSwaptionHW1D),
  {{"Price",DOUBLE,{100},FORBID},{" ",PREMIA_NULLTYPE,{0},
    FORBID}},
  CHK_OPT(CF_ReceiverSwaptionHW1D),
  CHK_ok,
  MET(Init)
} ;
```

References