```
   Help
#include <stdlib.h>
#include "sg1d_stdi.h"
#include "math/InterestRateModelTree/TreeShortRate/TreeSho
    rtRate.h"
#include "pnl/pnl_vector.h"
#include "math/read_market_zc/InitialYieldCurve.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)
int CALC(TR_BermudianSwaptionSG1D)(void *Opt,void *Mod,
    PricingMethod *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
static int CHK_OPT(TR_BermudianSwaptionSG1D)(void *Opt, voi
    d *Mod)
{
  return NONACTIVE;
}
#else

static void BermudianSwaption_InitialPayoffSG1D(TreeShortR
    ate* Meth, ModelParameters* ModelParam, PnlVect* ZCbondPric
    eVect1, PnlVect* ZCbondPriceVect2, PnlVect* OptionVect1, Pn
    lVect* OptionVect2, NumFunc_1 *p, double first_reset_date,
    double contract_maturity, double periodicity, double SwaptionFix
    edRate)
{
  double Ti2, Ti1, payoff;
  int i, j, i_Ti2, i_Ti1, n;
  int jminprev, jmaxprev;

  Ti2 = contract_maturity;
  Ti1 = Ti2 - periodicity;
  i_Ti1 = IndexTime(Meth, Ti1);

  jminprev = pnl_vect_int_get(Meth->Jminimum, Meth->Ngrid);
      // jmin(Ngrid)
```

```
jmaxprev = pnl_vect_int_get(Meth->Jmaximum, Meth->Ngrid);
    // jmax(Ngrid)

pnl_vect_resize(ZCbondPriceVect2, jmaxprev-jminprev+1);
pnl_vect_resize(OptionVect2, jmaxprev-jminprev+1);
pnl_vect_set_double(ZCbondPriceVect2, 1 + SwaptionFixedRa
  te * periodicity);

n = (int) ((contract_maturity-first_reset_date)/periodic
  ity + 0.1);

Ti1 = contract_maturity - periodicity; // Ti1 = T(n-1)
i_Ti1 = IndexTime(Meth, Ti1);

p->Par[0].Val.V_DOUBLE = 1.0 ;

BackwardIteration(Meth, ModelParam, ZCbondPriceVect1,     ZCbondPriceVect2, Me
  SwaptionFixedRate * periodicity) * P(T(n-1),T(n))

for( j = 0 ; j<ZCbondPriceVect1->size ; j++) // Price at
  T(n-1)
  {
    payoff = (p->Compute)(p->Par, GET(ZCbondPriceVect1,
  j));
    LET(OptionVect2, j) = payoff;
  }

for(i = n-2; i>=0; i--)
  {
    Ti1 = first_reset_date + i * periodicity; // Ti1 = T(
  i)
    Ti2 = Ti1 + periodicity;                  // Ti2 = T(
  i+1)

    i_Ti2 = IndexTime(Meth, Ti2);
    i_Ti1 = IndexTime(Meth, Ti1);

    pnl_vect_plus_double(ZCbondPriceVect2, SwaptionFixed
  Rate * periodicity);

    BackwardIteration(Meth, ModelParam, ZCbondPriceVect1,
```

```
     ZCbondPriceVect2, i_Ti2, i_Ti1, &func_model_sg1d); // {su
    m{j = i+1 à n} ci*P(Ti,Tj)

      BackwardIteration(Meth, ModelParam, OptionVect1,
    OptionVect2, i_Ti2, i_Ti1, &func_model_sg1d); // price of the
    swaption at T(i)


      for( j = 0 ; j<ZCbondPriceVect1->size ; j++)
        {
          payoff = (p->Compute)(p->Par, GET(ZCbondPriceVec
    t1, j)); // payoff at T(i)

          if(payoff > GET(OptionVect2,j))
            {
              LET(OptionVect2, j) = payoff;
              LET(OptionVect1, j) = payoff;
            }
        }
    }
}

/// Price of a Swaption using a trinomial tree

static double tr_sg1d_bermudianswaption(TreeShortRate*
    Meth, ModelParameters* ModelParam, ZCMarketData* ZCMarket,
    int NumberOfTimeStep, NumFunc_1 *p, double r, double periodic
    ity,double first_reset_date,double contract_maturity,
    double SwaptionFixedRate)
{
  double OptionPrice, Ti1;
  int i_Ti1;

  PnlVect* OptionPriceVect1; // Vector of prices of the
    option at i
  PnlVect* OptionPriceVect2; // Vector of prices of the
    option at i+1
  PnlVect* ZCbondPriceVect1; // Vector of prices of the
    option at i+1
  PnlVect* ZCbondPriceVect2; // Vector of prices of the
    option at i+1
```

```
  OptionPriceVect1 = pnl_vect_create(1);
  OptionPriceVect2 = pnl_vect_create(1);
  ZCbondPriceVect1 = pnl_vect_create(1);
  ZCbondPriceVect2 = pnl_vect_create(1);


  ///**************** PAYOFF at the MATURITY of the OPTION
    : T(0)****************///

  BermudianSwaption_InitialPayoffSG1D(Meth, ModelParam,     ZCbondPriceVect1, ZC
    eVect2, p, first_reset_date, contract_maturity, periodicity
    , SwaptionFixedRate);

  ///**************** Backward computation of the option
    price ****************///

  Ti1 = first_reset_date; // Ti1 = T(0)
  i_Ti1 = IndexTime(Meth, Ti1);

  BackwardIteration(Meth, ModelParam, OptionPriceVect1,
    OptionPriceVect2, i_Ti1, 0, &func_model_sg1d);

  OptionPrice = GET(OptionPriceVect1, 0);

  pnl_vect_free(& OptionPriceVect1);
  pnl_vect_free(& OptionPriceVect2);
  pnl_vect_free(& ZCbondPriceVect1);
  pnl_vect_free(& ZCbondPriceVect2);

  return OptionPrice;

}

static int tr_bermudianswaption1d(int flat_flag,double r0,
    double a,double sigma, double contract_maturity, double first_
    reset_date, double periodicity,double Nominal, double Swapt
    ionFixedRate, NumFunc_1 *p, long N_steps, double *price)
{
  TreeShortRate Tr;
  ModelParameters ModelParams;
  ZCMarketData ZCMarket;
```

```
/* Flag to decide to read or not ZC bond datas in "initia
  lyields.dat" */
/* If P(0,T) not read then P(0,T)=exp(-r0*T) */
if(flat_flag==0)
  {
    ZCMarket.FlatOrMarket = 0;
    ZCMarket.Rate = r0;
  }

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

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

ModelParams.MeanReversion  = a;
ModelParams.RateVolatility = sigma;

SetTimeGrid_Tenor(&Tr, N_steps, first_reset_date, contrac
  t_maturity, periodicity);

SetTreeShortRate(&Tr, &ModelParams, &ZCMarket, &func_
  model_sg1d, &func_model_der_sg1d, &func_model_inv_sg1d);

*price =  Nominal * tr_sg1d_bermudianswaption(&Tr, &
  ModelParams, &ZCMarket, N_steps, p, r0, periodicity, first_res
  et_date, contract_maturity, SwaptionFixedRate);

DeleteTreeShortRate(&Tr);
DeleteZCMarketData(&ZCMarket);

return OK;
```

```
}


///********************************************** PREMIA
    FUNCTIONS ***********************************************///

int CALC(TR_BermudianSwaptionSG1D)(void *Opt,void *Mod,
    PricingMethod *Met)
{
  TYPEOPT* ptOpt=(TYPEOPT*)Opt;
  TYPEMOD* ptMod=(TYPEMOD*)Mod;

  return  tr_bermudianswaption1d(ptMod->flat_flag.Val.V_
    INT,
                                 MOD(GetYield)(ptMod),
                                 ptMod->a.Val.V_DOUBLE,
                                 ptMod->Sigma.Val.V_PDOUB
    LE,
                                 ptOpt->BMaturity.Val.V_DA
    TE-ptMod->T.Val.V_DATE,
                                 ptOpt->OMaturity.Val.V_DA
    TE-ptMod->T.Val.V_DATE,
                                 ptOpt->ResetPeriod.Val.V_
    DATE,
                                 ptOpt->Nominal.Val.V_PDOUB
    LE,
                                 ptOpt->FixedRate.Val.V_PDO
    UBLE,
                                 ptOpt->PayOff.Val.V_
    NUMFUNC_1,
                                 Met->Par[0].Val.V_LONG,
                                 &(Met->Res[0].Val.V_
    DOUBLE));
}
static int CHK_OPT(TR_BermudianSwaptionSG1D)(void *Opt, voi
    d *Mod)
{
  if ((strcmp(((Option*)Opt)->Name,"PayerBermudanSwaption")
    ==0) || (strcmp(((Option*)Opt)->Name,"
    ReceiverBermudanSwaption")==0))
```

```
    return OK;
  else
    return WRONG;
}
#endif //PremiaCurrentVersion


static int MET(Init)(PricingMethod *Met,Option *Opt)
{
  if ( Met->init == 0)
    {
      Met->init=1;
      Met->HelpFilenameHint = "    tr_quadratic1d_bermudianswaption";

      Met->Par[0].Val.V_INT=50;
    }

  return OK;
}

PricingMethod MET(TR_BermudianSwaptionSG1D)=
{
  "TR_SquareGaussian1d_Swaption",
  {{"TimeStepNumber per Period",INT,{100},ALLOW},
      {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CALC(TR_BermudianSwaptionSG1D),
  {{"Price",DOUBLE,{100},FORBID} ,{" ",PREMIA_NULLTYPE,{0},
    FORBID}},
  CHK_OPT(TR_BermudianSwaptionSG1D),
  CHK_ok,
  MET(Init)
} ;
```

References