```
   Help
#include "varswap3d_std.h"
#include "enums.h"
#include "math/equity_pricer/implied_bs.h"
#include "pnl/pnl_finance.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2009+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(MC_VARSWAP3D)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(MC_VARSWAP3D)(void*Opt,void *Mod,PricingMethod *
    Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else

/**
 * State_vector at time i
 *
 * @param State_Vector : the matrix of gaussian numbers, mu
    st already be allocated
 * @param samples : number of Monte Carlo samples (= numb
    er of rows of M)
 * @param dimension : dimension of the simulation (= numb
    er of columns of M)
 * @param type_generator : index of the generator
 */
static void Euler_discretise(VARSWAP3D_MOD * M,
                             const double Delta_t,
                             PnlVect * State_Vector,
                             double * State_Vector_control,
                             double rhot,
                             double sigma,
                             double sqrt_delta_t,
                             int type_generator)
{
  int d;
```

```
  double sum,sigma_y,Inc1,Inc2;

  Inc1 =pnl_rand_normal(type_generator);//MGET(Increment_
    Vector,w,0);
  Inc2 =M->Rho*Inc1+
    +rhot*pnl_rand_normal(type_generator);

  sum=0.;
  for(d=1;d<State_Vector->size;d++)
    {
      sum+=GET(M->Beta,d-1)/GET(M->SqrtMeanReversion,d-1)*
    GET(State_Vector,d);
      LET(State_Vector,d)+=-GET(M->MeanReversion,d-1)*Delt
    a_t*GET(State_Vector,d)
        +GET(M->SqrtMeanReversion,d-1)*sqrt_delta_t*Inc2;
    }
  *State_Vector_control+=sigma*(-0.5*sigma+Inc1);
  sigma_y=sigma*exp(0.5*sum);
  LET(State_Vector,0)+=sigma_y*(-0.5*sigma_y+Inc1);

};

static void  Monte_Carlo_Solve_European(double * ptprice,
                                        double * ptdelta,
                                        VARSWAP3D_MOD * M,
                                        PnlVect * Initial_
    Value,
                                        int N_time,
                                        int samples,
                                        int type_generator)
{
  int w,d,i,dimension;
  double mean_price,mean_delta,var_price,var_delta,Delta_t;
  PnlVect *State_Vector;
  double price_traj,delta_traj,State_Vector_control,sqrt_de
    lta_t,rhot,sigma;

  dimension=M->Nb_factor+1;
  Delta_t=M->T/N_time;
  sqrt_delta_t=sqrt(Delta_t);
  rhot=sqrt(1-M->Rho*M->Rho);
```

```
sigma=sqrt_delta_t*M->V0;

mean_price=0;
mean_delta=0;
var_price=0;
var_delta=0;

State_Vector=pnl_vect_create(dimension);
// Solve SDE in all trajectories
pnl_rand_init(type_generator,2, samples);
for(w=0;w<samples;w++)
  {
    // ST= F_T^T   Ft^T = S_t exp((r-divd)(T-t))
    for(d=0;d<State_Vector->size;d++)
      LET(State_Vector,d)=GET(Initial_Value,d);
    State_Vector_control=GET(Initial_Value,0);
    for(i=0;i<N_time;i++)
      Euler_discretise(M,Delta_t,State_Vector,
                       &State_Vector_control,
                       rhot,sigma,sqrt_delta_t,
                       type_generator);

    price_traj=(MAX(M->F0*exp(GET(State_Vector,0))-M->
  Strike,0.0)
                -MAX(M->F0*exp(State_Vector_control)-M->
  Strike,0.0));
    if((M->is_call))
      delta_traj=((M->Strike<M->F0*exp(GET(State_Vector,0
  )))?1.0:0.0)
        -((M->Strike<M->F0*exp(State_Vector_control))?1.0
  :0.0);
    else
      delta_traj=((M->Strike>M->F0*exp(GET(State_Vector,0
  )))?-1.0:0.0)
        -((M->Strike>M->F0*exp(State_Vector_control))?-1.
  0:0.0);

    mean_price+=price_traj;
    mean_delta+=delta_traj;

    var_price+=SQR(price_traj);
```

```
      var_delta+=SQR(delta_traj);
    }
  *ptprice=M->Bond*mean_price/(double)samples;
  //error_price= M->Bond*sqrt(var_price/(double)samples-SQ
    R(*ptprice))/sqrt((double)samples);
  // Add BS price
  *ptprice+=pnl_bs_impli_call_put (M->is_call,M->V0,M->Bond
    ,M->F0,M->Strike,M->T);
  /* Delta estimator */
  *ptdelta=(mean_delta/(double)samples);

  //error_delta= sqrt((var_delta/(double)samples-SQR(*ptde
    lta)))/sqrt((double)samples);
  // Add BS price
  *ptdelta+=exp((M->R-M->Divid)*M->T)*pnl_bs_impli_call_
    put_delta_forward(M->is_call,M->V0,M->Bond,M->F0,M->Strike,M-
    >T);
  pnl_vect_free(&State_Vector);
}

static int mc_varswap3d(VARSWAP3D_MOD * M,PricingMethod *
    Met)
{
  long Drawing= Met->Par[0].Val.V_LONG;
  int N_T= Met->Par[1].Val.V_INT;
  int Generator=   Met->Par[2].Val.V_ENUM.value;
  PnlVect *Initial_Value=pnl_vect_create_from_zero(M->Nb_
    factor+1);
  Monte_Carlo_Solve_European(&(Met->Res[0].Val.V_DOUBLE),
                             &(Met->Res[1].Val.V_DOUBLE),
                             M,
                             Initial_Value,
                             N_T,
                             Drawing,
                             Generator);
  pnl_vect_free(&Initial_Value);
  return OK;
}

int CALC(MC_VARSWAP3D)(void *Opt, void *Mod, PricingMethod
    *Met)
```

```
{
  TYPEOPT* ptOpt=(TYPEOPT*)Opt;
  TYPEMOD* ptMod=(TYPEMOD*)Mod;
  double res;
  VARSWAP3D_MOD * M=svs_model_create_from_Model(ptMod);
  svs_model_initialise_from_Option(M,ptOpt);
  res=mc_varswap3d(M,Met);
  svs_model_free(&M);
  return res;
}

static int CHK_OPT(MC_VARSWAP3D)(void *Opt, void *Mod)
{
  if ((strcmp( ((Option*)Opt)->Name,"CallEuro")==0)||(strc
    mp( ((Option*)Opt)->Name,"PutEuro")==0))
    return OK;

  return  WRONG;
}

#endif //PremiaCurrentVersion
static int MET(Init)(PricingMethod *Met,Option *Opt)
{
  if ( Met->init == 0)
    {
      Met->init=1;
      Met->Par[0].Val.V_INT2=10000;
      Met->Par[1].Val.V_LONG=100;
      Met->Par[2].Val.V_ENUM.value=0;
      Met->Par[2].Val.V_ENUM.members=&PremiaEnumMCRNGs;
    }

  return OK;
}

PricingMethod MET(MC_VARSWAP3D)=
{
  "MC_VARSWAP3D",
  {{"Number of Iterations",INT2,{100},ALLOW},{"TimeStepNumb
    er",INT2,{100},ALLOW},{"RandomGenerator",ENUM,{100},ALLOW},{
    " ",PREMIA_NULLTYPE,{0},FORBID}},
```

```
  CALC(MC_VARSWAP3D),
  {{"Price",DOUBLE,{100},FORBID},{"Delta",DOUBLE,{100},FORB
    ID},
      {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CHK_OPT(MC_VARSWAP3D),
  CHK_ok,
  MET(Init)
};
```

References