```
   Help
#include  "bs1d_limdisc.h"
#define BIG_DOUBLE 1.0e6
#define INC 1.0e-5 /*Relative Increment for Delta-Hedging*/

/*TO BE DEVELOPED*/
int CALC(DynamicHedgingSimulator)(void *Opt,void *Mod,Prici
    ngMethod *Met,DynamicTest *Test)
{
  TYPEOPT* ptOpt=(TYPEOPT*)Opt;
  TYPEMOD* ptMod=(TYPEMOD*)Mod;
  int type_generator,error,init_mc;
  long path_number,hedge_number,i,j;
  double step_hedge,initial_stock,initial_time,stock,sell
    ing_price,delta,previous_delta;
  double cash_account,stock_account,cash_rate,stock_rate;
  double pl_sample,mean_pl,var_pl,min_pl,max_pl;
  double pl_sample_breached,mean_pl_breached,var_pl_breache
    d,
    min_pl_breached,max_pl_breached;
  double exp_trendxh,sigmaxsqrth;
  int     up,out,lim_breached,counter_breached;
  double  lim,r,divid,rebate,capit;

  up=(ptOpt->DownOrUp.Val.V_BOOL==UP);
  out=(ptOpt->OutOrIn.Val.V_BOOL==OUT);
  rebate=((ptOpt->Rebate.Val.V_NUMFUNC_1)->Compute)((ptOpt-
    >Rebate.Val.V_NUMFUNC_1)->Par,ptMod->T.Val.V_DATE);
  lim=((ptOpt->Limit.Val.V_NUMFUNC_1)->Compute)((ptOpt->    Limit.Val.V_NUMFUNC_


  initial_stock=ptMod->S0.Val.V_PDOUBLE;
  initial_time=ptMod->T.Val.V_DATE;

  type_generator=Test->Par[0].Val.V_INT;
  path_number=Test->Par[1].Val.V_LONG;
  hedge_number=Test->Par[2].Val.V_LONG;

  step_hedge=(ptOpt->Maturity.Val.V_DATE-ptMod->T.Val.V_DA
    TE)/(double)hedge_number;
```

```
r=log(1.+ptMod->R.Val.V_DOUBLE/100.);
divid=log(1.+ptMod->Divid.Val.V_DOUBLE/100.);
cash_rate=exp(r*step_hedge);
stock_rate=exp(divid*step_hedge)-1.;

sigmaxsqrth=ptMod->Sigma.Val.V_PDOUBLE*sqrt(step_hedge);
exp_trendxh=exp(ptMod->Mu.Val.V_DOUBLE*step_hedge-0.5*SQ
  R(sigmaxsqrth));

mean_pl=0.0;
var_pl=0.0;
min_pl=BIG_DOUBLE;
max_pl=-BIG_DOUBLE;

mean_pl_breached=0.0;
var_pl_breached=0.0;
min_pl_breached=BIG_DOUBLE;
max_pl_breached=-BIG_DOUBLE;

counter_breached=0;

init_mc= pnl_rand_init(type_generator,(int)hedge_number,
  path_number);
/* Test after initialization for the generator */
if(init_mc == OK)
  {
    for (i=0;i<path_number;i++)
      {

  ptMod->T.Val.V_DATE=initial_time;
  ptMod->S0.Val.V_PDOUBLE=initial_stock;
  if ((error=(Met->Compute)(Opt,Mod,Met)))
        {
      ptMod->T.Val.V_DATE=initial_time;
      ptMod->S0.Val.V_PDOUBLE=initial_stock;
      return error;
        };
  selling_price=Met->Res[0].Val.V_DOUBLE;
  delta=Met->Res[1].Val.V_DOUBLE;
  cash_account=selling_price-delta*initial_stock;
  stock_account=delta*initial_stock;
```

```
stock=initial_stock;
lim_breached=0;
capit=exp(r*(ptOpt->Maturity.Val.V_DATE-ptMod->T.Val.
V_DATE));

for (j=0;(j<hedge_number-1)&&(!out || !lim_breached);
j++)
        {
    ptMod->T.Val.V_DATE=ptMod->T.Val.V_DATE+step_hed
ge;

    previous_delta=delta;

    cash_account*=cash_rate;
    cash_account+=stock_rate*stock_account;
    capit=capit/cash_rate;
    stock*=exp_trendxh*exp(sigmaxsqrth*pnl_rand_normal
(type_generator));
    if (out)
            {
  if ((up && (stock>lim)) || (!up && (stock<lim)) )
                {
      counter_breached++;
      cash_account-=rebate;
      stock_account=delta*lim;

      pl_sample_breached=capit*(cash_account+stock_ac
count);
      mean_pl_breached=mean_pl_breached+pl_sample_br
eached;
      var_pl_breached=var_pl_breached+SQR(pl_sample_
breached);
      min_pl_breached=MIN(pl_sample_breached,min_pl_
breached);
      max_pl_breached=MAX(pl_sample_breached,max_pl_
breached);
      lim_breached=1;
                }
            }
    if (!out || !lim_breached)
```

```
            {
  ptMod->S0.Val.V_PDOUBLE=stock;
  if ((error=(Met->Compute)(Opt,Mod,Met)))
                {
      ptMod->T.Val.V_DATE=initial_time;
      ptMod->S0.Val.V_PDOUBLE=initial_stock;
      return error;
                };
  delta=Met->Res[1].Val.V_DOUBLE;

  cash_account-=(delta-previous_delta)*stock;
  stock_account=delta*stock;
            }
        }
if (!lim_breached) {
        cash_account*=cash_rate;
        cash_account+=stock_rate*stock_account;

        stock*=exp_trendxh*exp(sigmaxsqrth*pnl_rand_nor
mal(type_generator));
        if (out)
    {
            if ((up && (stock>lim)) || (!up && (stock<    lim)) )
  {
                counter_breached++;
                cash_account-=rebate;
                stock_account=delta*lim;

                pl_sample_breached=cash_account+stock_
account;
                mean_pl_breached=mean_pl_breached+pl_
sample_breached;
                var_pl_breached=var_pl_breached+SQR(pl_
sample_breached);
                min_pl_breached=MIN(pl_sample_breached,
min_pl_breached);
                max_pl_breached=MAX(pl_sample_breached,
max_pl_breached);

                lim_breached=1;
  }
```

```
    }

        if (!out || !lim_breached)
    {
            cash_account=cash_account-((double)(out ||     lim_breached))*((
Opt->PayOff.Val.V_NUMFUNC_1)->Par,stock)+delta*stock;

            pl_sample=cash_account;

            mean_pl=mean_pl+pl_sample;
            var_pl=var_pl+SQR(pl_sample);
            min_pl=MIN(pl_sample,min_pl);
            max_pl=MAX(pl_sample,max_pl);
    }
}/*!lim_breached*/

    }

  mean_pl=mean_pl/((double)(path_number-(long)counter_
breached));
  var_pl=var_pl/((double) (path_number-(long)counter_br
eached))-SQR(mean_pl);

  Test->Res[0].Val.V_DOUBLE=mean_pl;
  Test->Res[1].Val.V_DOUBLE=var_pl;
  Test->Res[2].Val.V_DOUBLE=min_pl;
  Test->Res[3].Val.V_DOUBLE=max_pl;
  if (counter_breached) {
    mean_pl_breached=mean_pl_breached/(double)counter_
breached;
    var_pl_breached=var_pl_breached/(double)counter_br
eached-SQR(mean_pl_breached);
  }

  Test->Res[4].Val.V_DOUBLE=mean_pl_breached;
  Test->Res[5].Val.V_DOUBLE=var_pl_breached;
  Test->Res[6].Val.V_DOUBLE=min_pl_breached;
  Test->Res[7].Val.V_DOUBLE=max_pl_breached;
  Test->Res[8].Val.V_LONG=(long)counter_breached;

  ptMod->T.Val.V_DATE=initial_time;
```

```
      ptMod->S0.Val.V_PDOUBLE=initial_stock;
      return OK;
    }
  else
    return init_mc;
}

static int TEST(Init)(DynamicTest *Test,Option *Opt)
{
  Test->Par[0].Val.V_INT=0;
  Test->Par[1].Val.V_LONG=1000;
  Test->Par[2].Val.V_LONG=250;

  return OK;
}
int CHK_TEST(test)(void *Opt, void *Mod, PricingMethod *
    Met)
{
  return OK;
}

DynamicTest MOD_OPT(test)=
{
  "Bs1dLimDyn{n",
  {{"Random Generator",INT,{100},ALLOW  },{"Path Number",
    LONG,{100},ALLOW},
   {"Hedge Number",LONG,{100},ALLOW},{" ",PREMIA_NULLTYPE,{
    0},FORBID}},
  CALC(DynamicHedgingSimulator),
  {{"Mean_P&l",DOUBLE,{100},FORBID},{"Var_P&l",DOUBLE,{100}
    ,FORBID},
   {"Min_P&l",DOUBLE,{100},FORBID},{"Max_P&l",DOUBLE,{100},
    FORBID},
   {"Mean_P&l_Breached",DOUBLE,{100},FORBID},{"Var_P&l_Brea
    ched",DOUBLE,{100},FORBID},
   {"Min_P&l_Breached",DOUBLE,{100},FORBID},{"Max_P&l_Brea
    ched",DOUBLE,{100},FORBID},
   {"Number_P&l_Breached",LONG,{100},FORBID},
   {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CHK_TEST(test),
  CHK_ok,
```

```
  TEST(Init)
};
```

References