```
   Help
#include <stdlib.h>
#include  "bs1d_lim.h"
#include "error_msg.h"
#define BIG_DOUBLE 1.0e6


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=0.,mean_pl,var_pl,min_pl,max_pl;
  double pl_sample_breached=0.,mean_pl_breached,var_pl_brea
    ched,
    min_pl_breached,max_pl_breached;
  double exp_trendxh,sigmaxsqrth;
  int     up,out,lim_breached,counter_breached;
  double  lim,r,divid,rebate,capit;

  /* Variables needed for exercise time of american options
     */
  int n_us;
  double sigma_us, /* Square deviation for the simulation
    of n_us */
    m_us;     /* Mean --- */

  /* Variables needed for Brownian bridge */
  double Bridge=0., d_Bridge, T1, BridgeT1, StockT1, H, si
    gma, mu;
  double currentT;

  /* Total or partial */
  int total;
  double starting_date, final_date, previous_stock=0., t_    capit;
```

```
/* Variables needed for Graphic outputs */
double *stock_array, *pl_array, *lim_array;
int  k, first, first_breached, j_breached=0;
double median_pl, median_pl_breached, current_mean_pl;
double current_date;
long size;

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_

/* Total or partial */
total=(ptOpt->PartOrTot.Val.V_BOOL==TOTAL);

starting_date=(ptOpt->Limit.Val.V_NUMFUNC_1)->Par[0].Val.
  V_DATE ;
final_date=(ptOpt->Limit.Val.V_NUMFUNC_1)->Par[1].Val.V_
  DATE ;

/******** Initialization of the test's parameters *******
  */
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;
current_date=ptMod->T.Val.V_DATE;

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;

init_mc=pnl_rand_init (type_generator,(int)hedge_number,
  path_number);
if (init_mc==OK) {

  counter_breached=0;

  /* Determining exercise time for american options */
  m_us=0.0;
  sigma_us=0.0;

  n_us=hedge_number;
  if ((ptOpt->EuOrAm.Val.V_BOOL==EURO) || (Test->Par[3].
  Val.V_BOOL == 0)) /* european */
    n_us=hedge_number;

  else if (Test->Par[3].Val.V_BOOL == 1) /* uniform on [0
  ,hedge_number] */
    n_us=(int)floor(pnl_rand_uni(type_generator)*(double)
  hedge_number)+1;

  else if (Test->Par[3].Val.V_BOOL == 2) /* "Integer"
  gaussian centered on the middle of [0,hedge_number] */
    {
      m_us=(int)floor(hedge_number/2.0);
      sigma_us=(int)floor(hedge_number/6.0);
      n_us=(int)floor(m_us+sigma_us*pnl_rand_normal(type_    generator))+1;
      if (n_us<0)
  n_us=0;
      else if (n_us>hedge_number)
```

```
n_us=hedge_number;
  };

/* Some initializations for Brownian Bridge */
sigma=ptMod->Sigma.Val.V_PDOUBLE;
mu=ptMod->Mu.Val.V_DOUBLE;
T1=Test->Par[6].Val.V_DATE-ptMod->T.Val.V_DATE;
StockT1=Test->Par[5].Val.V_PDOUBLE;
BridgeT1=(log(StockT1/initial_stock)-(mu-SQR(sigma)/2.0
)*T1)/sigma;

/* Graphic outputs initializations and dynamical memor
y allocutions */
first=1;
first_breached=1;
median_pl=0.0;
median_pl_breached=0.0;
size=hedge_number+1;

if ((stock_array= malloc(size*sizeof(double)))==NULL)
  return MEMORY_ALLOCATION_FAILURE;
if ((pl_array= malloc(size*sizeof(double)))==NULL)
  return MEMORY_ALLOCATION_FAILURE;
if ((lim_array= malloc(size*sizeof(double)))==NULL)
  return MEMORY_ALLOCATION_FAILURE;

for (k=10;k<=23;k++)
  {
    if (Test->Res[k].Val.V_PNLVECT != NULL)
      pnl_vect_resize (Test->Res[k].Val.V_PNLVECT, size
);
    else if ((Test->Res[k].Val.V_PNLVECT = pnl_vect_cr
eate (size))==NULL) /* Time */
      return MEMORY_ALLOCATION_FAILURE;
  }

if (Test->Res[24].Val.V_PNLVECT != NULL) pnl_vect_resiz
e (Test->Res[24].Val.V_PNLVECT, 2);
else if ((Test->Res[24].Val.V_PNLVECT=pnl_vect_create(2
))==NULL)
  return MEMORY_ALLOCATION_FAILURE;
```

```
  if (Test->Res[25].Val.V_PNLVECT != NULL) pnl_vect_resiz
  e (Test->Res[25].Val.V_PNLVECT, 2);
  else if ((Test->Res[25].Val.V_PNLVECT=pnl_vect_create(2
  ))==NULL) /* exercise Time */
    return MEMORY_ALLOCATION_FAILURE;

  for (k=0;k<=hedge_number;k++)
    Test->Res[10].Val.V_PNLVECT->array[k]=current_date+k*
  step_hedge;

  if (Test->Par[4].Val.V_BOOL==1)
    {
      Test->Res[24].Val.V_PNLVECT->array[0]=current_date+
  T1;
      Test->Res[24].Val.V_PNLVECT->array[1]=StockT1;
    }
  else
    {
      Test->Res[24].Val.V_PNLVECT->array[0]=current_date;
      Test->Res[24].Val.V_PNLVECT->array[1]=initial_stock
  ;
    }



  /******** Trajectories of the stock ********/
  for (i=0;i<path_number;i++)
    {
/* computing selling-price and delta */
      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;

/* computing cash_account and stock_account */
```

```
        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));

        stock_array[0]=stock;
        pl_array[0]=0;
        lim_array[0]=lim;

        /* Brownian bridge's initialization */
        if (Test->Par[4].Val.V_BOOL==1) /* With brownian br
    idge */
    {
            Bridge=0.0;
            H=0.0;
    }

  /******** Dynamic Hedge ********/
        for (j=1;(j<hedge_number)&&(!out || !lim_breached)&
    &(j<n_us);j++)
    {
            ptMod->T.Val.V_DATE=ptMod->T.Val.V_DATE+step_
    hedge;

            lim=((ptOpt->Limit.Val.V_NUMFUNC_1)->Compute)((
    ptOpt->Limit.Val.V_NUMFUNC_1)->Par,ptMod->T.Val.V_DATE);
            rebate=((ptOpt->Rebate.Val.V_NUMFUNC_1)->Compu
    te)((ptOpt->Rebate.Val.V_NUMFUNC_1)->Par,ptMod->T.Val.V_DA
    TE);


            previous_delta=delta;
            previous_stock=stock;

      /* Capitalization of cash_account and yielding divid
    ends */
            cash_account*=cash_rate;
            cash_account+=stock_rate*stock_account;
```

```
        capit=capit/cash_rate;

  /* computing the new stock's value */
        currentT=j*step_hedge;
        H=step_hedge/(T1-currentT);
        if ((T1>currentT)&&(H<=1)&&(Test->Par[4].Val.V_
BOOL==1)) /* Using Brownian Bridge */
    {
            d_Bridge=(BridgeT1-Bridge)*H+sqrt(step_hed
ge*(1-H))*pnl_rand_normal(type_generator);
            Bridge+=d_Bridge;
            stock*=exp_trendxh*exp(sigma*d_Bridge);
    }

        else /* After or without using Brownian Bridge
*/
    stock*=exp_trendxh*exp(sigmaxsqrth*pnl_rand_normal
(type_generator));

        if (out)
    {
            if ((total)||((!total)&&(currentT>=startin
g_date)&&(currentT<=final_date)))
  {
    /* If the stock has reached the limit */
                if ((up && (stock>lim))||(!up && (stock
<lim)))
      {
                    counter_breached++;
                    cash_account-=rebate;
                    if (Test->Par[7].Val.V_BOOL==0)
    stock_account=delta*lim;
                    else if (Test->Par[7].Val.V_BOOL==1
)
    stock_account=delta*stock;
                    else if (Test->Par[7].Val.V_BOOL==2
)
    {
                        stock_account=delta*lim;
                        t_capit=(lim-previous_stock)*
step_hedge/(stock-previous_stock);
```

```
                        t_capit=step_hedge-t_capit;
                        capit*=exp(r*t_capit);
    }
  /* computing and Capitalization of P&L */
                    pl_sample_breached=capit*(cash_ac
count+stock_account);
                    mean_pl_breached=mean_pl_breached+
pl_sample_breached;
                    var_pl_breached=var_pl_breached+SQ
R(pl_sample_breached);
                    min_pl_breached=MIN(pl_sample_brea
ched,min_pl_breached);
                    max_pl_breached=MAX(pl_sample_brea
ched,max_pl_breached);
                    lim_breached=1;

                    j_breached=j;
                    for (k=j_breached; k<=hedge_number;
 k++)
    {
                        pl_array[k]=pl_sample_breached;
                        stock_array[k]=stock;
                        lim_array[k]=lim;
    }
      }
  }
    }
  /* If the stock has not reached the limit */
        if (!out || !lim_breached)
    {
/* computing the new selling-price and the new delta
*/
            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;
```

```
    /* computing new cash_account and new stock_account *
    /
                cash_account-=(delta-previous_delta)*stock;
                stock_account=delta*stock;

                stock_array[j]=stock;
                pl_array[j]=cash_account-Met->Res[0].Val.V_
    DOUBLE+delta*stock;
                lim_array[j]=lim;
        }
    } /*j*/

  /******** Last hedge *******/
        if (!lim_breached)
    {
            lim=((ptOpt->Limit.Val.V_NUMFUNC_1)->Compute)((
    ptOpt->Limit.Val.V_NUMFUNC_1)->Par,ptMod->T.Val.V_DATE);
            rebate=((ptOpt->Rebate.Val.V_NUMFUNC_1)->Compu
    te)((ptOpt->Rebate.Val.V_NUMFUNC_1)->Par,ptMod->T.Val.V_DA
    TE);

      /* Capitalization of cash_account and yielding divid
    ends */
            cash_account*=cash_rate;
            cash_account+=stock_rate*stock_account;

      /* computing the last stock's value */
            currentT=j*step_hedge;
            H=step_hedge/(T1-currentT);
            if ((T1>currentT)&&(H<1)&&(Test->Par[4].Val.V_
    BOOL==1)) /* Using Brownian Bridge */
        {
                d_Bridge=(BridgeT1-Bridge)*H+sqrt(step_hed
    ge*(1-H))*pnl_rand_normal(type_generator);
                Bridge+=d_Bridge;
                stock*=exp_trendxh*exp(sigma*d_Bridge);
        }

            else /* After or without using Brownian Bridge
    */
        stock*=exp_trendxh*exp(sigmaxsqrth*pnl_rand_normal
```

```
(type_generator));

        if (out)
    {
            if ((total)||((!total)&&(currentT>=startin
g_date)&&(currentT<=final_date)))
  {
    /* If the stock has reached the limit */
                if ((up && (stock>lim))||(!up && (stock
<lim)))
      {
                    cash_account-=rebate;
                    if (Test->Par[7].Val.V_BOOL==0)
    stock_account=delta*lim;
                    else if (Test->Par[7].Val.V_BOOL==1
)
    stock_account=delta*stock;
                    else if (Test->Par[7].Val.V_BOOL==2
)
    {
                        stock_account=delta*lim;
                        t_capit=(lim-previous_stock)*
step_hedge/(stock-previous_stock);
                        t_capit=step_hedge-t_capit;
                        capit*=exp(r*t_capit);
    }
  /* computing and Capitalization of P&L */
                    pl_sample_breached=capit*(cash_ac
count+stock_account);
                    mean_pl_breached=mean_pl_breached+
pl_sample_breached;
                    var_pl_breached=var_pl_breached+SQ
R(pl_sample_breached);
                    min_pl_breached=MIN(pl_sample_brea
ched,min_pl_breached);
                    max_pl_breached=MAX(pl_sample_brea
ched,max_pl_breached);

                    lim_breached=1;
                    counter_breached++;
```

```
                        j_breached=j;
                        pl_array[j_breached]=pl_sample_brea
  ched;
                        stock_array[j_breached]=stock;

          }
      }
        }
      /* If the stock has not reached the limit */
            if (!out || !lim_breached)
        {
  /* Capitalization of cash_account and computing the
  P&L using the PayOff*/
              cash_account=cash_account-((double)(out ||     lim_breached))*((
  Opt->PayOff.Val.V_NUMFUNC_1)->Par,stock)+delta*stock;
              pl_sample=capit*cash_account;

              stock_array[hedge_number]=stock;
              pl_array[hedge_number]=pl_sample;
              lim_array[hedge_number]=lim;

              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*/

if (((lim_breached)&&(n_us<j_breached)&&(n_us<hedge_numb
  er))||((!lim_breached)&&(n_us<hedge_number)))
  {
    for (k=n_us; k<=hedge_number; k++)
      {
  pl_array[k]=pl_array[n_us-1];
  stock_array[k]=stock_array[n_us-1];
  lim_array[k]=lim_array[n_us-1];
      }
  }

/* Selection of trajectories (Spot and P&L) for graphic
  outputs */
```

```
if (!lim_breached)
  {
    if (first)
      {

  for (k=0; k<=hedge_number; k++)
    {
      Test->Res[11].Val.V_PNLVECT->array[k]=stock_arra
  y[k];
      Test->Res[12].Val.V_PNLVECT->array[k]=stock_arra
  y[k];
      Test->Res[13].Val.V_PNLVECT->array[k]=stock_arra
  y[k];
      Test->Res[14].Val.V_PNLVECT->array[k]=pl_array[k]
  ;
      Test->Res[15].Val.V_PNLVECT->array[k]=pl_array[k]
  ;
      Test->Res[16].Val.V_PNLVECT->array[k]=pl_array[k]
  ;
    }
  first=0;
  median_pl=pl_sample;
      }
    else
      {
  current_mean_pl=mean_pl/i;
  if (pl_sample==min_pl)
    {
      for (k=0; k<=hedge_number; k++)
        {
    Test->Res[11].Val.V_PNLVECT->array[k]=stock_array[
  k];
    Test->Res[14].Val.V_PNLVECT->array[k]=pl_array[k];
        }
    }
  else if (pl_sample==max_pl)
    {
      for (k=0; k<=hedge_number; k++)
        {
    Test->Res[12].Val.V_PNLVECT->array[k]=stock_array[
  k];
```

```
    Test->Res[15].Val.V_PNLVECT->array[k]=pl_array[k];
        }
    }
  else if (SQR(pl_sample-current_mean_pl) < SQR(median_
  pl-current_mean_pl))
    {
      median_pl=pl_sample;
      for (k=0; k<=hedge_number; k++)
        {
    Test->Res[13].Val.V_PNLVECT->array[k]=stock_array[
  k];
    Test->Res[16].Val.V_PNLVECT->array[k]=pl_array[k];
        }
    }
      }
  } /*!lim_breached*/
else
  {
    if (first_breached)
      {

  for (k=0; k<=hedge_number; k++)
    {
      Test->Res[17].Val.V_PNLVECT->array[k]=stock_arra
  y[k];
      Test->Res[18].Val.V_PNLVECT->array[k]=stock_arra
  y[k];
      Test->Res[19].Val.V_PNLVECT->array[k]=stock_arra
  y[k];
      Test->Res[20].Val.V_PNLVECT->array[k]=pl_array[k]
  ;
      Test->Res[21].Val.V_PNLVECT->array[k]=pl_array[k]
  ;
      Test->Res[22].Val.V_PNLVECT->array[k]=pl_array[k]
  ;
    }
  first_breached=0;
  median_pl_breached=pl_sample_breached;
      }
    else
      {
```

```
current_mean_pl=mean_pl_breached/i;
if (pl_sample_breached==min_pl_breached)
  {
    for (k=0; k<=hedge_number; k++)
      {
  Test->Res[17].Val.V_PNLVECT->array[k]=stock_array[
k];
  Test->Res[20].Val.V_PNLVECT->array[k]=pl_array[k];
      }
  }
else if (pl_sample_breached==max_pl_breached)
  {
    for (k=0; k<=hedge_number; k++)
      {
  Test->Res[18].Val.V_PNLVECT->array[k]=stock_array[
k];
  Test->Res[21].Val.V_PNLVECT->array[k]=pl_array[k];
      }
  }
else if (SQR(pl_sample_breached-current_mean_pl) < SQ
R(median_pl_breached-current_mean_pl))
  {
    median_pl_breached=pl_sample_breached;
    for (k=0; k<=hedge_number; k++)
      {
  Test->Res[19].Val.V_PNLVECT->array[k]=stock_array[
k];
  Test->Res[22].Val.V_PNLVECT->array[k]=pl_array[k];
      }
  }
    }
}
  } /*i*/


for (k=0; k<=hedge_number; k++)
  {
    Test->Res[23].Val.V_PNLVECT->array[k]=lim_array[k];
  }

Test->Res[25].Val.V_PNLVECT->array[0]=current_date+n_us
```

```
  *step_hedge;
  Test->Res[25].Val.V_PNLVECT->array[1]=initial_stock;

  mean_pl=mean_pl/((double)(path_number-(long)counter_br
  eached));
  var_pl=var_pl/((double) (path_number-(long)counter_brea
  ched))-SQR(mean_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);
    }

  if (first)
    {
mean_pl=0.;
var_pl=0.;
min_pl=0.;
max_pl=0.;
for (k=0; k<=hedge_number; k++)
  {
          Test->Res[11].Val.V_PNLVECT->array[k]=initial_
  stock;
          Test->Res[12].Val.V_PNLVECT->array[k]=initial_
  stock;
          Test->Res[13].Val.V_PNLVECT->array[k]=initial_
  stock;
          Test->Res[14].Val.V_PNLVECT->array[k]=0.;
          Test->Res[15].Val.V_PNLVECT->array[k]=0.;
          Test->Res[16].Val.V_PNLVECT->array[k]=0.;
  }
    }
  if (first_breached)
    {
mean_pl_breached=0.;
var_pl_breached=0.;
min_pl_breached=0.;
max_pl_breached=0.;
```

```
  for (k=0; k<=hedge_number; k++)
    {
      Test->Res[17].Val.V_PNLVECT->array[k]=initial_stock;
      Test->Res[18].Val.V_PNLVECT->array[k]=initial_stock;
      Test->Res[19].Val.V_PNLVECT->array[k]=initial_stock;
      Test->Res[20].Val.V_PNLVECT->array[k]=0.;
      Test->Res[21].Val.V_PNLVECT->array[k]=0.;
      Test->Res[22].Val.V_PNLVECT->array[k]=0.;
    }
      }

    free(stock_array);
    free(pl_array);
    free(lim_array);

    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;

    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;

    Test->Res[9].Val.V_DOUBLE=current_date+n_us*step_hedge;

    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)
{
  static int first=1;
  TYPEOPT* pt=(TYPEOPT*)(Opt->TypeOpt);
```

```
  if (first)
    {

      Test->Par[0].Val.V_INT=0;        /* Random Generator *
    /
      Test->Par[1].Val.V_LONG=1000;    /* PathNumber */
      Test->Par[2].Val.V_LONG=250;     /* HedgeNumber */
      Test->Par[3].Val.V_BOOL=0;       /* exerciseType */
      Test->Par[4].Val.V_BOOL=1;       /* Brownian Bridge */
      Test->Par[5].Val.V_PDOUBLE=90.; /* SpotTarget */
      Test->Par[6].Val.V_DATE=0.5;     /* TimeTarget */
      Test->Par[7].Val.V_BOOL=2;       /* LimReachedMethod *
    /

      Test->Res[10].Val.V_PNLVECT = NULL;
      Test->Res[11].Val.V_PNLVECT = NULL;
      Test->Res[12].Val.V_PNLVECT = NULL;
      Test->Res[13].Val.V_PNLVECT = NULL;
      Test->Res[14].Val.V_PNLVECT = NULL;
      Test->Res[15].Val.V_PNLVECT = NULL;
      Test->Res[16].Val.V_PNLVECT = NULL;
      Test->Res[17].Val.V_PNLVECT = NULL;
      Test->Res[18].Val.V_PNLVECT = NULL;
      Test->Res[19].Val.V_PNLVECT = NULL;
      Test->Res[20].Val.V_PNLVECT = NULL;
      Test->Res[21].Val.V_PNLVECT = NULL;
      Test->Res[22].Val.V_PNLVECT = NULL;
      Test->Res[23].Val.V_PNLVECT = NULL;
      Test->Res[24].Val.V_PNLVECT = NULL;
      Test->Res[25].Val.V_PNLVECT = NULL;

      first=0;
    }

  if (pt->EuOrAm.Val.V_INT==EURO)
    Test->Par[3].Viter=IRRELEVANT;

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

```
{
  return OK;
}

DynamicTest MOD_OPT(test)=
{
  "bs1d_lim_test",
  {{"Random Generator",INT,{100},ALLOW},
   {"Path Number",LONG,{100},ALLOW},
   {"Hedge Number",LONG,{100},ALLOW},
   {"exerciseType",BOOL,{100},ALLOW},        /* 0: european;
     1: american "uniform"; 2: american "gaussian" */
   {"BrownianBridge",BOOL,{100},ALLOW},      /* 0: without
    brownian bridge; 1: with brownian bridge */
   {"SpotTarget",PDOUBLE,{100},ALLOW},
   {"TimeTarget",DATE,{100},ALLOW},
   {"LimReachedMethod",BOOL,{100},ALLOW},    /* if lim rea
    ched,   0: delta*lim at currentT;
                 1: delta*stock at currentT;
                 2: delta*lim at "linear time" */
   {" ",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_Breached",DOUBLE,{100},FORBID},
   {"Min_P&l_Breached",DOUBLE,{100},FORBID},
   {"Max_P&l_Breached",DOUBLE,{100},FORBID},
   {"Number_P&l_Breached",LONG,{100},FORBID},
   {"exerciseTime",DOUBLE,{100},FORBID},


   {"Time",PNLVECT,{100},FORBID},
   {"Stockmin",PNLVECT,{0},FORBID},
   {"Stockmax",PNLVECT,{0},FORBID},
   {"Stockmean",PNLVECT,{0},FORBID},
   {"PLmin",PNLVECT,{0},FORBID},
   {"PLmax",PNLVECT,{0},FORBID},
```

```
    {"PLmean",PNLVECT,{0},FORBID},
    {"Stockminbreached",PNLVECT,{0},FORBID},
    {"Stockmaxbreached",PNLVECT,{0},FORBID},
    {"Stockmeanbreached",PNLVECT,{0},FORBID},
    {"PLminbreached",PNLVECT,{0},FORBID},
    {"PLmaxbreached",PNLVECT,{0},FORBID},
    {"PLmeanbreached",PNLVECT,{0},FORBID},
    {"LimitBarrier",PNLVECT,{0},FORBID},
    {"SpotTarget",PNLVECT,{0},FORBID},
    {"exerciseTime",PNLVECT,{0},FORBID},

    {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CHK_TEST(test),
  CHK_ok,
  TEST(Init)
};
```

References