

[Help](#)

```
#include "bs1d_lim.h"

static DoubleArray ttime=
{
    0,NULL
};
static DoubleArray stockmin=
{
    0,NULL
};
static DoubleArray stockmax=
{
    0,NULL
};
static DoubleArray stockmean=
{
    0,NULL
};
static DoubleArray plmin=
{
    0,NULL
};
static DoubleArray plmax=
{
    0,NULL
};
static DoubleArray plmean=
{
    0,NULL
};
static DoubleArray stockminbreached=
{
    0,NULL
};
static DoubleArray stockmaxbreached=
{
    0,NULL
};
static DoubleArray stockmeanbreached=
{
```

```
    0,NULL
};
static DoubleArray plminbreached=
{
    0,NULL
};
static DoubleArray plmaxbreached=
{
    0,NULL
};
static DoubleArray plmeanbreached=
{
    0,NULL
};

static DoubleArray limit=
{
    0,NULL
};
static DoubleArray target=
{
    0,NULL
};
static DoubleArray exercise=
{
    0,NULL
};

int CALC(DynamicHedgingSimulator)(void *Opt,void
    *Mod,PricingMethod *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,selling_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_breached,
        min_pl_breached,max_pl_breached;
double exp_trendxh,sigmaxsqqrth;
int      up,out,lim_breached,counter_breached;
double   lim,r,divid,rebate,capit;

/* Variables needed for exercise time of am
erican 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, d_Bridge, T1, BridgeT1, Stock
T1, H, sigma, mu;
double currentT;

/* Total or partial */
int total;
double starting_date, final_date, previous_
stock, t_capit;

/* Variables needed for Graphic outputs */
double *stock_array, *pl_array, *lim_array;
int k, first, first_breached, j_breached;
double median_pl, median_pl_breached, curren
t_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_1)->Par,ptMod->T.

```

```

Val.V_DATE);

/* 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 para
meters *****/
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_DATE)/(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*SQR(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=InitGenerator (type_generator,(int)hed
ge_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) /* uni
form on [0,hedge_number] */
    n_us=(int)floor(Uniform(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*Gauss_Abram
owitzStegun(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_PDDOUBLE;
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 memory allocations */
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_DOUBLEARRAY->array=malloc(size*sizeof(double)))==NULL)
return MEMORY_ALLOCATION_FAILURE;
else
Test->Res[k].Val.V_DOUBLEARRAY->size=size;
}

if ((Test->Res[24].Val.V_DOUBLEARRAY->array=malloc(2*sizeof(double)))==NULL) /* Target */
return MEMORY_ALLOCATION_FAILURE;
else
Test->Res[24].Val.V_DOUBLEARRAY->size=2;
if ((Test->Res[25].Val.V_DOUBLEARRAY->array=malloc(2*sizeof(double)))==NULL) /* exercise

```

```

Time */
return MEMORY_ALLOCATION_FAILURE;
else
    Test->Res[25].Val.V_DOUBLEARRAY->size=2;

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

if (Test->Par[4].Val.V_BOOL==1)
{
    Test->Res[24].Val.V_DOUBLEARRAY->array[0]
=current_date+T1;
    Test->Res[24].Val.V_DOUBLEARRAY->array[1]
=StockT1;
}
else
{
    Test->Res[24].Val.V_DOUBLEARRAY->array[0]
=current_date;
    Test->Res[24].Val.V_DOUBLEARRAY->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 bridge */
    {
        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_DA
TE+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)->Compute)((ptOpt->Rebate.Val.V_NUMFUNC_1)->
Par,ptMod->T.Val.V_DATE);

        previous_delta=delta;

```



```

        previous_stock=stock;

        /* Capitalization of cash_account and y
        ielding dividends */
        cash_account*=cash_rate;
        cash_account+=stock_rate*stock_accou
nt;
        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_hedge*(1-H))*Gauss_AbramowitzStegun(type_g
enerator);
            Bridge+=d_Bridge;
            stock*=exp_trendxh*exp(sigma*d_Br
idge);
        }

        else /* After or without using Browni
an Bridge */
            stock*=exp_trendxh*exp(sigmamaxsqrt
h*Gauss_AbramowitzStegun(type_generator));

        if (out)
        {
            if ((total)||(!total)&&(currentT
>=starting_date)&&(currentT<=final_date)))
            {
                /* If the stock has reached the
limit */
                if ((up && (stock>lim))||(!u
p && (stock<lim)))
                {
                    counter_breached++;
                    cash_account-=rebate;
                    if (Test-&gtPar[7].Val.V_BO

```

```

OL==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
s_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_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;

                                j_breached=j;
                                for (k=j_breached; k<=hed
ge_number; k++)
                                {
                                pl_array[k]=pl_sample
_breached;

                                stock_array[k]=stock;
                                lim_array[k]=lim;

```

```

    }
    }
    }
    }
    /* If the stock has not reached the limit
    t */
    if (!out || !lim_breached)
    {
        /* computing the new selling-price
        and the new delta */
        ptMod->S0.Val.V_PDDOUBLE=stock;
        if (error=(Met->Compute)(Opt,Mod,
        Met))
        {
            ptMod->T.Val.V_DATE=initial_
            time;
            ptMod->S0.Val.V_PDDOUBLE=ini
            tial_stock;
            return error;
        };
        delta=Met->Res[1].Val.V_DOUBLE;

        /* computing new cash_account and ne
        w stock_account */
        cash_account-=(delta-previous_de
        lta)*stock;
        stock_account=delta*stock;

        stock_array[j]=stock;
        pl_array[j]=cash_account-Met->Re
        s[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)->Compute)((ptOpt->Rebate.Val.V_NUMFUNC_1)->
Par,ptMod->T.Val.V_DATE);

        /* Capitalization of cash_account and y
ielding dividends */
        cash_account*=cash_rate;
        cash_account+=stock_rate*stock_accou
nt;

        /* 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_hedge*(1-H))*Gauss_AbramowitzStegun(type_g
enerator);
            Bridge+=d_Bridge;
            stock*=exp_trendxh*exp(sigma*d_Br
idge);
        }

        else /* After or without using Browni
an Bridge */
            stock*=exp_trendxh*exp(sigmamaxsqrt
h*Gauss_AbramowitzStegun(type_generator));

        if (out)
        {
            if ((total)||(!total)&&(currentT
>=starting_date)&&(currentT<=final_date)))
            {
                /* If the stock has reached the
limit */
                if ((up && (stock>lim))||(!u
p && (stock<lim)))
                {
                    cash_account-=rebate;
                    if (Test->Par[7].Val.V_BO

```

```

OL==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
s_stock)*step_hedge/(stock-previous_stock);
    t_capit=step_hedge-t_
capit;
    capit*=exp(r*t_capit)
;
}
/* computing and Capitalizatio
n of P&L */
pl_sample_breached=capit*
(cash_account+stock_account);
mean_pl_breached=mean_pl_
breached+pl_sample_breached;
var_pl_breached=var_pl_br
eached+SQR(pl_sample_breached);
min_pl_breached=MIN(pl_sa
mple_breached,min_pl_breached);
max_pl_breached=MAX(pl_sa
mple_breached,max_pl_breached);

lim_breached=1;
counter_breached++;

j_breached=j;
pl_array[j_breached]=pl_
sample_breached;
stock_array[j_breached]=
stock;

```

```

    }
    }
}

/* If the stock has not reached the limit
t */
if (!out || !lim_breached)
{
    /* Capitalization of cash_account
and computing the P&L using the PayOff*/
    cash_account=cash_account-((
double)(out || lim_breached))*((ptOpt->PayOff.Val.V_
NUMFUNC_1)->Compute)((ptOpt->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_number)))||((!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_
DOUBLEARRAY->array[k]=stock_array[k];
                Test->Res[12].Val.V_
DOUBLEARRAY->array[k]=stock_array[k];
                Test->Res[13].Val.V_
DOUBLEARRAY->array[k]=stock_array[k];
                Test->Res[14].Val.V_
DOUBLEARRAY->array[k]=pl_array[k];
                Test->Res[15].Val.V_
DOUBLEARRAY->array[k]=pl_array[k];
                Test->Res[16].Val.V_
DOUBLEARRAY->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_numb
er; k++)
                {
                    Test->Res[11].Val.V_
DOUBLEARRAY->array[k]=stock_array[k];
                    Test->Res[14].Val.V_
DOUBLEARRAY->array[k]=pl_array[k];
                }
            }
            else if (pl_sample==max_pl)
            {

```



```

DOUBLEARRAY->array[k]=pl_array[k];
    Test->Res[22].Val.V_
DOUBLEARRAY->array[k]=pl_array[k];
    }
    first_breached=0;
    median_pl_breached=pl_sample_
breached;
    }
    else
    {
        current_mean_pl=mean_pl_brea
ched/i;
        if (pl_sample_breached==min_
pl_breached)
        {
            for (k=0; k<=hedge_numb
er; k++)
            {
                Test->Res[17].Val.V_
DOUBLEARRAY->array[k]=stock_array[k];
                Test->Res[20].Val.V_
DOUBLEARRAY->array[k]=pl_array[k];
            }
        }
        else if (pl_sample_breached==
max_pl_breached)
        {
            for (k=0; k<=hedge_numb
er; k++)
            {
                Test->Res[18].Val.V_
DOUBLEARRAY->array[k]=stock_array[k];
                Test->Res[21].Val.V_
DOUBLEARRAY->array[k]=pl_array[k];
            }
        }
        else if (SQR(pl_sample_brea
ched-current_mean_pl) < SQR(median_pl_breached-cu
rrent_mean_pl))
        {
            median_pl_breached=pl_sa

```

```

mple_breached;
        for (k=0; k<=hedge_number; k++)
        {
            Test->Res[19].Val.V_
DOUBLEARRAY->array[k]=stock_array[k];
            Test->Res[22].Val.V_
DOUBLEARRAY->array[k]=pl_array[k];
        }
    }
}
} /*i*/

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

Test->Res[25].Val.V_DOUBLEARRAY->array[0]=current_date+n_us*step_hedge;
Test->Res[25].Val.V_DOUBLEARRAY->array[1]=initial_stock;

mean_pl=mean_pl/((double)(path_number-(long)
counter_breached));
var_pl=var_pl/((double) (path_number-(long)
counter_breached))-SQR(mean_pl);

if (counter_breached)
{
    mean_pl_breached=mean_pl_breached/(
double)counter_breached;
    var_pl_breached=var_pl_breached/(double)
counter_breached-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_DOUBLEARRAY->array[k]=initial_stock;
    Test->Res[12].Val.V_DOUBLEARRAY->array[k]=initial_stock;
    Test->Res[13].Val.V_DOUBLEARRAY->array[k]=initial_stock;
    Test->Res[14].Val.V_DOUBLEARRAY->array[k]=0.;
    Test->Res[15].Val.V_DOUBLEARRAY->array[k]=0.;
    Test->Res[16].Val.V_DOUBLEARRAY->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_DOUBLEARRAY->array[k]=initial_stock;
        Test->Res[18].Val.V_DOUBLEARRAY->array[k]=initial_stock;
        Test->Res[19].Val.V_DOUBLEARRAY->array[k]=initial_stock;
        Test->Res[20].Val.V_DOUBLEARRAY->array[k]=0.;
        Test->Res[21].Val.V_DOUBLEARRAY->array[k]=0.;
        Test->Res[22].Val.V_DOUBLEARRAY->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 Gen
        erator */
        Test->Par[1].Val.V_LONG=1000;      /* PathNumb

```

```

er */
Test->Par[2].Val.V_LONG=250;    /* HedgeNumb
er */
Test->Par[3].Val.V_BOOL=0;      /* exerciseTy
pe */
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;      /* LimReache
dMethod */

Test->Res[10].Val.V_DOUBLEARRAY=&ttime;
Test->Res[11].Val.V_DOUBLEARRAY=&stockmin;
Test->Res[12].Val.V_DOUBLEARRAY=&stockmax;
Test->Res[13].Val.V_DOUBLEARRAY=&stockmean;
Test->Res[14].Val.V_DOUBLEARRAY=&plmin;
Test->Res[15].Val.V_DOUBLEARRAY=&plmax;
Test->Res[16].Val.V_DOUBLEARRAY=&plmean;
Test->Res[17].Val.V_DOUBLEARRAY=&stockminbre
ached;
Test->Res[18].Val.V_DOUBLEARRAY=&stockmaxbrea
ched;
Test->Res[19].Val.V_DOUBLEARRAY=&stockmeanbre
ached;
Test->Res[20].Val.V_DOUBLEARRAY=&plminbreac
hed;
Test->Res[21].Val.V_DOUBLEARRAY=&plmaxbreache
d;
Test->Res[22].Val.V_DOUBLEARRAY=&plmeanbreac
hed;
Test->Res[23].Val.V_DOUBLEARRAY=&limit;
Test->Res[24].Val.V_DOUBLEARRAY=&target;
Test->Res[25].Val.V_DOUBLEARRAY=&exercise;

first=0;
}

if (pt->EuOrAm.Val.V_INT==EURO)

```

```

        Test->Par[3].Viter=IRRELEVANT;

    return OK;
}

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

DynamicTest MOD_OPT(test)=
{
    "bsld_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 "gau
    ssian" */
    {"BrownianBridge",BOOL,100,ALLOW},        /* 0:
    without brownian bridge; 1: with brownian bridg
    e */
    {"SpotTarget",PDOUBLE,100,ALLOW},
    {"TimeTarget",DATE,100,ALLOW},
    {"LimReachedMethod",BOOL,100,ALLOW},      /* if
    lim reached,    0: delta*lim at currentT;

                1: delta*stock at currentT;

                2: delta*lim at "linear time" */
    {" ",END,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",DOUBLEARRAY,100,FORBID},
{"Stockmin",DOUBLEARRAY,0,FORBID},
{"Stockmax",DOUBLEARRAY,0,FORBID},
{"Stockmean",DOUBLEARRAY,0,FORBID},
{"PLmin",DOUBLEARRAY,0,FORBID},
{"PLmax",DOUBLEARRAY,0,FORBID},
{"PLmean",DOUBLEARRAY,0,FORBID},
{"Stockminbreached",DOUBLEARRAY,0,FORBID},
{"Stockmaxbreached",DOUBLEARRAY,0,FORBID},
{"Stockmeanbreached",DOUBLEARRAY,0,FORBID},
{"PLminbreached",DOUBLEARRAY,0,FORBID},
{"PLmaxbreached",DOUBLEARRAY,0,FORBID},
{"PLmeanbreached",DOUBLEARRAY,0,FORBID},
{"LimitBarrier",DOUBLEARRAY,0,FORBID},
{"SpotTarget",DOUBLEARRAY,0,FORBID},
{"exerciseTime",DOUBLEARRAY,0,FORBID},

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

```

References