

[Help](#)

```
#include "bs1d_pad.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 padmin=
{
    0,NULL
};
static DoubleArray padmax=
{
    0,NULL
};
static DoubleArray padmean=
{
```

```

    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,path_dep,ini
        tial_path_dep,initial_time,stock,selling_price,de
        lta,previous_delta;
    double cash_account,stock_account,cash_rate,
        stock_rate;
    double pl_sample,mean_pl,var_pl,min_pl,max_pl;
    double exp_muxh,sigmaxsqsrth;
    double r,divid;

    /* 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, StockT1,
        H, sigma, mu; /* Brownian bridge */
    double currentT;

    /* Variables needed for Graphic outputs */
    double *stock_array, *pl_array, *pad_array, cu

```

```

    rrent_mean_pl, median_pl;
int    k;
long size;
double current_date;

/***** Initialization of the test's parameters *****/
initial_stock=ptMod->S0.Val.V_PDOUBLE;
initial_path_dep=(ptOpt->PathDep.Val.V_NUMFUNC_2)->Par[3].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_muxh=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;

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

```

```

    /* 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_Abramow
        itzStegun(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_DA
    TE;
    StockT1=Test->Par[5].Val.V_PDOUBLE;
    BridgeT1=(log(StockT1/initial_stock)-(mu-SQR(
    sigma)/2.0)*T1)/sigma;

    /* Graphic outputs initializations and dynami
    cal memory allocutions */
    current_mean_pl=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 ((pad_array=malloc(size*sizeof(double)))==
NULL)
    return MEMORY_ALLOCATION_FAILURE;

for (k=5;k<=14;k++)
{
    if ((Test->Res[k].Val.V_DOUBLEARRAY->array=
malloc(size*sizeof(double)))==NULL) /* Time */
        return MEMORY_ALLOCATION_FAILURE;
    else
        Test->Res[k].Val.V_DOUBLEARRAY->size=siz
e;
}

if ((Test->Res[15].Val.V_DOUBLEARRAY->array=
malloc(2*sizeof(double)))==NULL) /* Target */
    return MEMORY_ALLOCATION_FAILURE;
else
    Test->Res[15].Val.V_DOUBLEARRAY->size=2;
if ((Test->Res[16].Val.V_DOUBLEARRAY->array=
malloc(2*sizeof(double)))==NULL) /* exercise Time
*/
    return MEMORY_ALLOCATION_FAILURE;
else
    Test->Res[16].Val.V_DOUBLEARRAY->size=2;

for (k=0;k<=hedge_number;k++)
    Test->Res[5].Val.V_DOUBLEARRAY->array[k]=cu
rrent_date+k*step_hedge;

if (Test->Par[4].Val.V_BOOL==1)
{

```

```

Test->Res[15].Val.V_DOUBLEARRAY->array[0]=
current_date+T1;
Test->Res[15].Val.V_DOUBLEARRAY->array[1]=
StockT1;
}
else
{
Test->Res[15].Val.V_DOUBLEARRAY->array[0]=
current_date;
Test->Res[15].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;
(ptOpt->PathDep.Val.V_NUMFUNC_2)->Par[4].
Val.V_PDOUBLE=initial_path_dep;
if (error=(Met->Compute)(Opt,Mod,Met))
{
ptMod->T.Val.V_DATE=initial_time;
ptMod->S0.Val.V_PDOUBLE=initial_stock;
(ptOpt->PathDep.Val.V_NUMFUNC_2)->Par[4]
.Val.V_PDOUBLE= initial_path_dep;
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;
path_dep=initial_path_dep;

```

```

stock_array[0]=stock;
pl_array[0]=0;
pad_array[0]=path_dep;

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

/***** Dynamic Hedge *****/
for (j=1;(j<hedge_number) && (j<n_us);j++)
{
    ptMod->T.Val.V_DATE=ptMod->T.Val.V_DATE+
step_hedge;
    previous_delta=delta;

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

    /* computing the new stock's value */
    currentT=j*step_hedge;
    H=step_hedge/(T1-currentT);
    if ((currentT<T1)&&(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_genera
tor);
        Bridge+=d_Bridge;
        stock*=exp_muxh*exp(sigma*d_Bridge);
    }
    else /* After or without using Brownian
Bridge */
        stock*=exp_muxh*exp(sigmamaxsrth*Gauss

```

```

_AbramowitzStegun(type_generator));

    /* computing the new selling-price and
the new delta */
    path_dep=((ptOpt->PathDep.Val.V_NUMFUNC_
2)->Compute)((ptOpt->PathDep.Val.V_NUMFUNC_2)->
Par,stock,ptMod->T.Val.V_DATE);
    ptMod->S0.Val.V_PDOUBLE=stock;
    (ptOpt->PathDep.Val.V_NUMFUNC_2)->Par[4]
.Val.V_PDOUBLE=path_dep;
    if (error=(Met->Compute)(Opt,Mod,Met))
    {
        ptMod->T.Val.V_DATE=initial_time;
        ptMod->S0.Val.V_PDOUBLE=initial_stock
;
        (ptOpt->PathDep.Val.V_NUMFUNC_2)->
Par[4].Val.V_PDOUBLE=initial_path_dep;
        return error;
    };
    delta=Met->Res[1].Val.V_DOUBLE;

    /* computing new cash_account and new
stock_account */
    cash_account+=(delta-previous_delta)*sto
ck;
    stock_account=delta*stock;

    stock_array[j]=stock;
    pl_array[j]=cash_account-Met->Res[0].Val
.V_DOUBLE+delta*stock;
    pad_array[j]=path_dep;
} /*j*/

/***** Last hedge *****/
ptMod->T.Val.V_DATE=ptMod->T.Val.V_DATE+ste
p_hedge;

/* Capitalization of cash_account and yield
ing dividends */
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 ((currentT<T1)&&(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_generator
);
    Bridge+=d_Bridge;
    stock*=exp_muxh*exp(sigma*d_Bridge);
}
else /* After or without using Brownian Br
idge */
    stock*=exp_muxh*exp(sigmamaxsqrth*Gauss_Ab
ramowitzStegun(type_generator));

path_dep=((ptOpt->PathDep.Val.V_NUMFUNC_2)-
>Compute)((ptOpt->PathDep.Val.V_NUMFUNC_2)->Par,
stock,ptMod->T.Val.V_DATE);

/* Capitalization of cash_account and compu
ting the P&L using the PayOff*/
cash_account=cash_account-((ptOpt->PayOff.
Val.V_NUMFUNC_2)->Compute)((ptOpt->PayOff.Val.V_
NUMFUNC_2)->Par,stock,path_dep)+delta*stock;
pl_sample=cash_account*exp((hedge_number-n_
us)*log(cash_rate));

if (n_us<hedge_number)
    for (k=n_us;k<=hedge_number;k++)
    {
        stock_array[k]=stock_array[n_us-1];
        pl_array[k]=pl_array[n_us-1];
        pad_array[k]=pad_array[n_us-1];
    }
else
{
    stock_array[hedge_number]=stock;

```

```

    pl_array[hedge_number]=pl_sample;
    pad_array[hedge_number]=path_dep;
}

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);

    /* Selection of trajectories (Spot and P&
L) for graphic outputs */
if (i==0)
{
    for (k=0; k<=hedge_number; k++)
    {
        Test->Res[6].Val.V_DOUBLEARRAY->array
[k]=stock_array[k];
        Test->Res[7].Val.V_DOUBLEARRAY->array
[k]=stock_array[k];
        Test->Res[8].Val.V_DOUBLEARRAY->array
[k]=stock_array[k];
        Test->Res[9].Val.V_DOUBLEARRAY->array
[k]=pl_array[k];
        Test->Res[10].Val.V_DOUBLEARRAY->ar
ray[k]=pl_array[k];
        Test->Res[11].Val.V_DOUBLEARRAY->ar
ray[k]=pl_array[k];
        Test->Res[12].Val.V_DOUBLEARRAY->ar
ray[k]=pad_array[k];
        Test->Res[13].Val.V_DOUBLEARRAY->ar
ray[k]=pad_array[k];
        Test->Res[14].Val.V_DOUBLEARRAY->ar
ray[k]=pad_array[k];

    }
    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[6].Val.V_DOUBLEARRAY->array[k]=stock_array[k];
            Test->Res[9].Val.V_DOUBLEARRAY->array[k]=pl_array[k];
            Test->Res[12].Val.V_DOUBLEARRAY->array[k]=pad_array[k];
        }
    }
    else if (pl_sample==max_pl)
    {
        for (k=0; k<=hedge_number; k++)
        {
            Test->Res[7].Val.V_DOUBLEARRAY->array[k]=stock_array[k];
            Test->Res[10].Val.V_DOUBLEARRAY->array[k]=pl_array[k];
            Test->Res[13].Val.V_DOUBLEARRAY->array[k]=pad_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[8].Val.V_DOUBLEARRAY->array[k]=stock_array[k];
            Test->Res[11].Val.V_DOUBLEARRAY->array[k]=pl_array[k];
            Test->Res[14].Val.V_DOUBLEARRAY->array[k]=pad_array[k];
        }
    }
}

} /*i*/

```

```

Test->Res[16].Val.V_DOUBLEARRAY->array[0]=current_date+n_us*step_hedge;
Test->Res[16].Val.V_DOUBLEARRAY->array[1]=initial_stock;

free(stock_array);
free(pl_array);
free(pad_array);

mean_pl=mean_pl/(double)path_number;
var_pl=var_pl/(double)path_number-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;
Test->Res[4].Val.V_DOUBLE=current_date+n_us*
step_hedge;

ptMod->T.Val.V_DATE=initial_time;
ptMod->S0.Val.V_PDOUBLE=initial_stock;
(ptOpt->PathDep.Val.V_NUMFUNC_2)->Par[4].Val.
V_PDOUBLE=initial_path_dep;

return 0;
}
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 G

```

```

        enerator */
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;      /* exercis
    eType */
Test->Par[4].Val.V_BOOL=1;      /* Browni
    an Bridge */
Test->Par[5].Val.V_PDDOUBLE=90.; /* SpotTa
    rget */
Test->Par[6].Val.V_DATE=0.5;    /* TimeTarg
    et */

Test->Res[5].Val.V_DOUBLEARRAY=&ttime;
Test->Res[6].Val.V_DOUBLEARRAY=&stockmin;

Test->Res[7].Val.V_DOUBLEARRAY=&stockmax;

Test->Res[8].Val.V_DOUBLEARRAY=&stockmean;
Test->Res[9].Val.V_DOUBLEARRAY=&plmin;
Test->Res[10].Val.V_DOUBLEARRAY=&plmax;
Test->Res[11].Val.V_DOUBLEARRAY=&plmean;
Test->Res[12].Val.V_DOUBLEARRAY=&padmin;
Test->Res[13].Val.V_DOUBLEARRAY=&padmin;
Test->Res[14].Val.V_DOUBLEARRAY=&padmin;
Test->Res[15].Val.V_DOUBLEARRAY=&target;
Test->Res[16].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)=
{
    "bs1d_pad_test",
    {"RandomGenerator",INT,100,ALLOW},
    {"PathNumber",LONG,100,ALLOW},
    {"HedgeNumber",LONG,100,ALLOW},
    {"exerciseType",BOOL,100,ALLOW},          /* 0:
        european; 1: american "uniform"; 2: american "gau
        ssian" */
    {"BrownianBridge",BOOL,100,ALLOW},        /* 0: w
        ithout brownian bridge; 1: with brownian bridge */
    {"SpotTarget",PDOUBLE,100,ALLOW},
    {"TimeTarget",DATE,100,ALLOW},

    {" ",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},
    {"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},
    {"PADmin",DOUBLEARRAY,0,FORBID},
    {"PADmax",DOUBLEARRAY,0,FORBID},
    {"PADmean",DOUBLEARRAY,0,FORBID},
    {"SpotTarget",DOUBLEARRAY,0,FORBID},
    {"exerciseTime",DOUBLEARRAY,0,FORBID},

    {" ",END,0,FORBID}},
    CHK_TEST(test),

```

```
CHK_ok,  
TEST(Init)  
};
```

References