```
   Help
#include  "bs1d_std.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 hedgetimemin=
    {
    0,NULL
    };
static DoubleArray hedgespotmin=
    {
    0,NULL
    };
static DoubleArray hedgetimemax=
    {
```

```
    0,NULL
    };
static DoubleArray hedgespotmax=
    {
    0,NULL
    };
static DoubleArray hedgetimemean=
    {
    0,NULL
    };
static DoubleArray hedgespotmean=
    {
    0,NULL
    };
static DoubleArray deltamin=
    {
    0,NULL
    };
static DoubleArray deltamax=
    {
    0,NULL
    };
static DoubleArray deltamean=
    {
    0,NULL
    };

int CALC(DynamicHedgingSimulatorPatry5)(void *
    Opt,void *Mod,PricingMethod *Met,DynamicTest *Test)
    {
    TYPEOPT* ptOpt=(TYPEOPT*)Opt;
    TYPEMOD* ptMod=(TYPEMOD*)Mod;
    int  type_generator,error;
    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 exp_trendxh,sigmaxsqrth;
```

```
  double r,divid;
double temp,pl_temp,deltaoptimal;
int indicehedge,ii;
int nbcouv;
double sumnbcouv;

  /* Variables needed for Graphic outputs */
  double *stock_array, *pl_array,*hedge_time, *
  hedge_spot, current_mean_pl, median_pl;
  double *delta_array;
int  k;
  long size;
  double current_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;

  InitGenerator (type_generator,1,path_number);

  /* 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 ((hedge_time=malloc(size*sizeof(double)))==
  NULL)
  return MEMORY_ALLOCATION_FAILURE;
  if ((hedge_spot=malloc(size*sizeof(double)))=
  =NULL)
  return MEMORY_ALLOCATION_FAILURE;
if ((delta_array=malloc(size*sizeof(double)))=
  =NULL)
  return MEMORY_ALLOCATION_FAILURE;

  for (k=9;k<=24;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=siz
  e;
  }


  for (k=0;k<=hedge_number;k++) /* Time */
      Test->Res[9].Val.V_DOUBLEARRAY->array[k]=
  current_date+k*step_hedge;

sumnbcouv=0.0;
```

```
  /******** 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;
    stock_array[0]=stock;
    pl_array[0]=0.0;
pl_temp=0.0;
delta_array[0]=delta;
hedge_time[0]=ptMod->T.Val.V_DATE;
hedge_spot[0]=stock;
indicehedge=1;

/******** Dynamic Hedge ********/
for (j=1;(j<hedge_number) ;j++)
        {
        previous_delta=delta;

  /* Capitalization of cash_account and y
ielding dividends */
```

```
        cash_account*=cash_rate;
        cash_account+=stock_rate*stock_accou
nt;

  stock*=exp_trendxh*exp(sigmaxsqrth*Gauss
_AbramowitzStegun(type_generator));

  /* computing the new selling-price and
the new delta */
        ptMod->T.Val.V_DATE=ptMod->T.Val.V_DA
TE+step_hedge;
        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;
            };

  deltaoptimal=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;

  temp=fabs((pl_array[j]-pl_temp)/pl_temp)
;

  if (temp>Test->Par[3].Val.V_DOUBLE)
    {delta=deltaoptimal;
```

```
    cash_account-=(delta-previous_delta)*
stock;
    stock_account=delta*stock;
    hedge_time[indicehedge]=ptMod->T.Val.
V_DATE;
    hedge_spot[indicehedge]=stock;
    pl_temp=pl_array[j];
    indicehedge++;
    }
  delta_array[j]=delta;

        } /*j*/

nbcouv=indicehedge;
sumnbcouv+=nbcouv;


for (ii=indicehedge;ii<=hedge_number;ii++)
{hedge_time[ii]=hedge_time[ii-1];
hedge_spot[ii]=hedge_spot[ii-1];
}


/******** Last hedge *******/
/* Capitalization of cash_account and yield
ing dividends */
    cash_account*=cash_rate;
    cash_account+=stock_rate*stock_account;

/* Computing the stock's last value */

    stock*=exp_trendxh*exp(sigmaxsqrth*Gauss_
AbramowitzStegun(type_generator));


/* Capitalization of cash_account and compu
ting the P&L using the PayOff*/
    cash_account=cash_account-((ptOpt->PayOf
f.Val.V_NUMFUNC_1)->Compute)((ptOpt->PayOff.Val.
V_NUMFUNC_1)->Par,stock)+delta*stock;
    pl_sample=cash_account;
```

```
  stock_array[hedge_number]=stock;
pl_array[hedge_number]=pl_sample;
  delta_array[hedge_number]=delta;


    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[10].Val.V_DOUBLEARRAY->
array[k]=stock_array[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]=pl_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->ar
ray[k]=delta_array[k];
    Test->Res[17].Val.V_DOUBLEARRAY->ar
ray[k]=delta_array[k];
    Test->Res[18].Val.V_DOUBLEARRAY->ar
ray[k]=delta_array[k];
            Test->Res[19].Val.V_DOUBLEARRAY->
array[k]=hedge_time[k];
            Test->Res[20].Val.V_DOUBLEARRAY->
array[k]=hedge_spot[k];
            Test->Res[21].Val.V_DOUBLEARRAY->
```

```
array[k]=hedge_time[k];
            Test->Res[22].Val.V_DOUBLEARRAY->
array[k]=hedge_spot[k];
            Test->Res[23].Val.V_DOUBLEARRAY->
array[k]=hedge_time[k];
            Test->Res[24].Val.V_DOUBLEARRAY->
array[k]=hedge_spot[k];
  }
  Test->Res[5].Val.V_INT=nbcouv;
  Test->Res[6].Val.V_INT=nbcouv;
  Test->Res[7].Val.V_INT=nbcouv;
        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[10].Val.V_DOUBLEARR
AY->array[k]=stock_array[k];
                Test->Res[13].Val.V_DOUBLEARR
AY->array[k]=pl_array[k];
                Test->Res[16].Val.V_DOUBLEARR
AY->array[k]=delta_array[k];
      Test->Res[19].Val.V_DOUBLEARRAY->
array[k]=hedge_time[k];
      Test->Res[20].Val.V_DOUBLEARRAY->
array[k]=hedge_spot[k];
      }
    Test->Res[5].Val.V_INT=nbcouv;
            }
        else if (pl_sample==max_pl)
            {
            for (k=0; k<=hedge_number; k++)
                {
                Test->Res[11].Val.V_DOUBLEARR
AY->array[k]=stock_array[k];
                Test->Res[14].Val.V_DOUBLEARR
AY->array[k]=pl_array[k];
```

```
                Test->Res[17].Val.V_DOUBLEARR
  AY->array[k]=delta_array[k];
        Test->Res[21].Val.V_DOUBLEARRAY->
  array[k]=hedge_time[k];
        Test->Res[22].Val.V_DOUBLEARRAY->
  array[k]=hedge_spot[k];
        }
      Test->Res[6].Val.V_INT=nbcouv;
             }
          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[12].Val.V_DOUBLEARR
  AY->array[k]=stock_array[k];
                 Test->Res[15].Val.V_DOUBLEARR
  AY->array[k]=pl_array[k];
                 Test->Res[18].Val.V_DOUBLEARR
  AY->array[k]=delta_array[k];
        Test->Res[23].Val.V_DOUBLEARRAY->
  array[k]=hedge_time[k];
        Test->Res[24].Val.V_DOUBLEARRAY->
  array[k]=hedge_spot[k];
        }
      Test->Res[7].Val.V_INT=nbcouv;
             }
          }
      } /*i*/

  Test->Res[8].Val.V_DOUBLE=sumnbcouv/(double)
  Test->Par[1].Val.V_LONG;

  free(stock_array);
  free(pl_array);
free(hedge_time);
free(hedge_spot);
free(delta_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=median_pl;

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

    return OK;
    }


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;       /* PathNu
    mber */
    Test->Par[2].Val.V_LONG=250;        /* HedgeN
    umber */
  Test->Par[3].Val.V_DOUBLE=0.1;    /* P&L_Targ
    et */
  Test->Par[4].Vtype=END;

    Test->Res[9].Val.V_DOUBLEARRAY=&ttime;
    Test->Res[10].Val.V_DOUBLEARRAY=&stockmin;
    Test->Res[11].Val.V_DOUBLEARRAY=&stockmax;
    Test->Res[12].Val.V_DOUBLEARRAY=&stockmean;
    Test->Res[13].Val.V_DOUBLEARRAY=&plmin;
    Test->Res[14].Val.V_DOUBLEARRAY=&plmax;
    Test->Res[15].Val.V_DOUBLEARRAY=&plmean;
```

```
    Test->Res[16].Val.V_DOUBLEARRAY=&deltamin;
  Test->Res[17].Val.V_DOUBLEARRAY=&deltamax;
  Test->Res[18].Val.V_DOUBLEARRAY=&deltamean;
    Test->Res[19].Val.V_DOUBLEARRAY=&hedgetimem
    in;
    Test->Res[20].Val.V_DOUBLEARRAY=&hedgespotm
    in;
    Test->Res[21].Val.V_DOUBLEARRAY=&hedgetimemax
    ;
    Test->Res[22].Val.V_DOUBLEARRAY=&hedgespotmax
    ;
    Test->Res[23].Val.V_DOUBLEARRAY=&hedgetimemea
    n;
    Test->Res[24].Val.V_DOUBLEARRAY=&hedgespotmea
    n;

  Test->Res[25].Vtype=END;

    first=0;
    }



    return OK;
    }
int CHK_TEST(test2)(void *Opt, void *Mod, Pricing
    Method *Met)
    {
  if ( (strcmp( Met->Name,"TR_PatryMartini")==0)
     || (strcmp( Met->Name,"TR_PatryMartini1")==0))
    return WRONG;
  else
    return OK;
    }

DynamicTest MOD_OPT(test2)=
    {
    "bs1d_std_test2",

    {{"RandomGenerator",INT,100,ALLOW},
    {"PathNumber",LONG,100,ALLOW},
```

```
  {"HedgeNumber",LONG,100,ALLOW},
{"P&L_Target",DOUBLE,0,ALLOW},
  {" ",END,0,FORBID}},

  CALC(DynamicHedgingSimulatorPatry5),

  {{"Mean_P&l",DOUBLE,100,FORBID},
  {"Var_P&l",DOUBLE,100,FORBID},
  {"Min_P&l",DOUBLE,100,FORBID},
  {"Max_P&l",DOUBLE,100,FORBID},
  {"Median_P&l",DOUBLE,100,FORBID},
{"NbHedgemin",INT,100,FORBID},
{"NbHedgemax",INT,100,FORBID},
{"NbHedgemean",INT,100,FORBID},
{"Mean of Number hedging",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},
  {"deltamin",DOUBLEARRAY,0,FORBID},
{"deltamax",DOUBLEARRAY,0,FORBID},
{"deltamean",DOUBLEARRAY,0,FORBID},
  {"HedgeTimemin",DOUBLEARRAY,0,FORBID},
  {"HedgeSpotmin",DOUBLEARRAY,0,FORBID},
  {"HedgeTimemax",DOUBLEARRAY,0,FORBID},
  {"HedgeSpotmax",DOUBLEARRAY,0,FORBID},
  {"HedgeTimemean",DOUBLEARRAY,0,FORBID},
  {"HedgeSpotmean",DOUBLEARRAY,0,FORBID},
  {" ",END,0,FORBID}},
CHK_TEST(test2),
  CHK_ok,
  TEST(Init)
  };
```

References