```
   Help
#include  "bs1d_std.h"

static double *Mesh=NULL, *Path=NULL, *Price=NUL
    L, *VectInvMeshDensity=NULL;
static double *Aux_BS_TD_1=NULL, *Aux_BS_TD_2=NUL
    L, *InvSigma=NULL;
static double Norm_BS_TD,DetInvSigma;
static double *AuxBS=NULL,*Sigma=NULL,*Aux_Stock=
    NULL;

static int BrGl_Allocation(long AL_Mesh_Size,
               int OP_Exercise_Dates, int
    BS_Dimension)
{
  int dummy;

  if (Mesh==NULL)
  Mesh=(double*)malloc(AL_Mesh_Size*OP_Exercise_
    Dates*BS_Dimension*sizeof(double));
  if (Mesh==NULL)
  return MEMORY_ALLOCATION_FAILURE;

  if (Price==NULL)
  Price=(double*)malloc(AL_Mesh_Size*OP_Exercis
    e_Dates*sizeof(double));
  if (Price==NULL)
  return MEMORY_ALLOCATION_FAILURE;

  if (Path==NULL)
  Path=(double*)malloc(OP_Exercise_Dates*BS_Dim
    ension*sizeof(double));

  if (Path==NULL)
  return MEMORY_ALLOCATION_FAILURE;

  if (VectInvMeshDensity==NULL)
  VectInvMeshDensity=(double*)malloc(AL_Mesh_Siz
    e*sizeof(double));

  if (VectInvMeshDensity==NULL)
```

```
return MEMORY_ALLOCATION_FAILURE;

if (Aux_BS_TD_1==NULL)
Aux_BS_TD_1=(double*)malloc(BS_Dimension*sizeo
  f(double));
if (Aux_BS_TD_1==NULL)
return MEMORY_ALLOCATION_FAILURE;

if (Aux_BS_TD_2==NULL)
Aux_BS_TD_2=(double*)malloc(BS_Dimension*sizeo
  f(double));
if (Aux_BS_TD_2==NULL)
return MEMORY_ALLOCATION_FAILURE;

if (InvSigma==NULL)
InvSigma=(double*)malloc(BS_Dimension*BS_Dimen
  sion*sizeof(double));
if (InvSigma==NULL)
return MEMORY_ALLOCATION_FAILURE;

if (Sigma==NULL)
Sigma=(double*)malloc(BS_Dimension*BS_Dimensi
  on*sizeof(double));
if (Sigma==NULL)
return MEMORY_ALLOCATION_FAILURE;

if (AuxBS==NULL)
AuxBS=(double*)malloc(BS_Dimension*sizeof(
  double));
if (AuxBS==NULL)
return MEMORY_ALLOCATION_FAILURE;

if (Aux_Stock==NULL)
Aux_Stock=(double*)malloc(BS_Dimension*sizeof(
  double));
if (Aux_Stock==NULL)
return MEMORY_ALLOCATION_FAILURE;


dummy=Chol_Allocation(BS_Dimension);
```

```
  return OK;
}

static void Brod_Liberation()
{
  if (Mesh!=NULL) {
  free(Mesh);
  Mesh=NULL;
  }

  if (Price!=NULL) {
  free(Price);
  Price=NULL;
  }

  if (Path!=NULL) {
  free(Path);
  Path=NULL;
  }

  if (VectInvMeshDensity!=NULL) {
  free(VectInvMeshDensity);
  VectInvMeshDensity=NULL;
  }

  if (Aux_BS_TD_1!=NULL) {
  free(Aux_BS_TD_1);
  Aux_BS_TD_1=NULL;
  }

  if (Aux_BS_TD_2!=NULL) {
  free(Aux_BS_TD_2);
  Aux_BS_TD_2=NULL;
  }

  if (InvSigma!=NULL) {
  free(InvSigma);
  InvSigma=NULL;
  }
  if (Aux_Stock!=NULL) {
  free(Aux_Stock);
```

```
  Aux_Stock=NULL;
  }

  if (AuxBS!=NULL) {
  free(AuxBS);
  AuxBS=NULL;
  }

  if (Sigma!=NULL) {
  free(Sigma);
  Sigma=NULL;
  }
  Chol_Liberation();
}

static double Discount(double Time, double BS_
    Interest_Rate)
{
  return exp(-BS_Interest_Rate*Time);
}

static void BS_Transition_Allocation(int BS_Dimen
    sion, double Step)
{
  int i,j;

  for (i=0;i<BS_Dimension;i++)
  for (j=0;j<BS_Dimension;j++)
    InvSigma[i*BS_Dimension+j]=Sigma[i*BS_Dimen
    sion+j];

  InverseCholesky(InvSigma,BS_Dimension);

  /* determinant of InvSigma */
  DetInvSigma=1;
  for (i=0;i<BS_Dimension;i++)
  DetInvSigma*=InvSigma[i*BS_Dimension+i];


  Norm_BS_TD=exp(-BS_Dimension*0.5*log(2.*PI*Ste
    p));
```

```
}

/*Black-Sholes Conditional Density function know
    ing Z*/
static double BS_TD(double *X, double *Z, int BS_
    Dimension, double Step)
{
  int i,j;
  double aux1,aux2;

  for (i=0;i<BS_Dimension;i++){
  Aux_BS_TD_1[i]=log(Z[i]/X[i])+Step*AuxBS[i];
  }
  aux1=Z[0];
  for (i=1;i<BS_Dimension;i++){
  aux1*=Z[i];
  }
  if (aux1==0){
  return -1;
  }
  else {
  for (i=0;i<BS_Dimension;i++){
    Aux_BS_TD_2[i]=0;
    for (j=0;j<=i;j++){
    Aux_BS_TD_2[i]+=InvSigma[i*BS_Dimension+j]*
    Aux_BS_TD_1[j];
    }
  }
  aux2=0;
  for (i=0;i<BS_Dimension;i++){
    aux2+=Aux_BS_TD_2[i]*Aux_BS_TD_2[i];
  }
  aux2=exp(-aux2/(2.*Step));
  return Norm_BS_TD*DetInvSigma*aux2/aux1;
  }
}

static double MeshDensity(int Time, double *Stock
    , int OP_Exercise_Dates, int AL_Mesh_Size,int
    BS_Dimension, double *BS_Spot, double Step)
{
```

```
  long k;
  double aux=0;

  if (Time>1){
  for (k=0;k<AL_Mesh_Size;k++)
    aux+=BS_TD(Mesh+k*OP_Exercise_Dates*BS_Dimen
    sion+(Time-1)*BS_Dimension,Stock,BS_Dimension,Ste
    p);
  return aux/(double)AL_Mesh_Size;
  } else {
  return BS_TD(BS_Spot,Stock,BS_Dimension,Step);
  }
}

static double Weight(int Time, double *iStock,
    double *jStock, int j, int BS_Dimension,
           double Step)
{
  if (Time>0){
  return BS_TD(iStock,jStock,BS_Dimension,Step)*
    VectInvMeshDensity[j];
  } else {
  return 1.;
  }
}

/*Black-Sholes Step*/
static void BS_Forward_Step(int mc_or_qmc,int gen
    erator,double *Stock, double *Initial_Stock, int
    BS_Dimension,double Step,double Sqrt_Step)
{
  int j,k;
  double Aux;

  for (j=0;j<BS_Dimension;j++){
  Aux_Stock[j]=Sqrt_Step*Gaussians[mc_or_qmc](1,
     CREATE, 0, generator);

  }
  for (j=0;j<BS_Dimension;j++){
  Aux=0.;
```

```
  for (k=0;k<=j;k++){
    Aux+=Sigma[j*BS_Dimension+k]*Aux_Stock[k];
  }
  Aux-=Step*AuxBS[j];
  Stock[j]=Initial_Stock[j]*exp(Aux);
  }
}

static void InitMesh(int mc_or_qmc,int generator,
    int AL_Mesh_Size, int BS_Dimension, double *BS_Spot
    ,int OP_Exercise_Dates, double Step, double Sqrt
    _Step)
{
  int j,k, aux;


  for (k=0;k<AL_Mesh_Size;k++){
  BS_Forward_Step(mc_or_qmc,generator,Mesh+k*OP_
    Exercise_Dates*BS_Dimension+BS_Dimension,BS_Spot,
    BS_Dimension,Step,Sqrt_Step);
  }
  for (j=2;j<OP_Exercise_Dates;j++){
  for (k=0;k<AL_Mesh_Size;k++){

      aux=(int) (AL_Mesh_Size*Uniform(generator))
    ;

    BS_Forward_Step(mc_or_qmc,generator,Mesh+k*
    OP_Exercise_Dates*BS_Dimension+j*BS_Dimension,Mes
    h+aux*OP_Exercise_Dates*BS_Dimension+(j-1)*BS_Dim
    ension,BS_Dimension,Step,Sqrt_Step);
  }
  }
}

static void BrGl(double *AL_Price,
         long AL_MonteCarlo_Iterations,
         NumFunc_1 *p, int AL_Mesh_Size,
         int AL_ShuttingDown,
         int mc_or_qmc,int generator,
         int OP_Exercise_Dates,
```

```
          double *BS_Spot,
          double BS_Maturity,
          double BS_Interest_Rate,
          double *BS_Dividend_Rate,
          double *BS_Volatility,
          int gj_flag)
{
  double aux,Step,Sqrt_Step,DiscountStep;
  long i,j,k;
  int l;
  double AL_BPrice,AL_FPrice;
  int BS_Dimension=1;


  AL_BPrice=0.;
  AL_FPrice=0.;
  Step=BS_Maturity/(double)(OP_Exercise_Dates-1);
  Sqrt_Step=sqrt(Step);
  DiscountStep=exp(-BS_Interest_Rate*Step);

  /*Memory Allocation*/
  BrGl_Allocation(AL_Mesh_Size,OP_Exercise_Dates,
    BS_Dimension);

  /*Black-Sholes initalization parameters*/
  Sigma[0]=BS_Volatility[0];
  BS_Transition_Allocation(BS_Dimension,Step);
  AuxBS[0]=0.5*SQR(BS_Volatility[0])-BS_Interest_
    Rate+BS_Dividend_Rate[0];

  /*Initialization of the mesh*/
  InitMesh(mc_or_qmc,generator,AL_Mesh_Size,BS_
    Dimension,BS_Spot,OP_Exercise_Dates,Step,Sqrt_Ste
    p);


  for (i=0;i<AL_Mesh_Size;i++)
  Price[i*OP_Exercise_Dates+OP_Exercise_Dates-1]
    =0.;

  /* Dynamical programing: Backward Price */
```

```
for (j=OP_Exercise_Dates-2;j>=1;j--){
  for (i=0;i<AL_Mesh_Size;i++){
  VectInvMeshDensity[i]=1./MeshDensity(j+1,Mes
  h+i*OP_Exercise_Dates*BS_Dimension+(j+1)*BS_Dimen
  sion,OP_Exercise_Dates,AL_Mesh_Size,BS_Dimension,
  BS_Spot,Step);

}
for (i=0;i<AL_Mesh_Size;i++){
  aux=0;
  for (k=0;k<AL_Mesh_Size;k++){

  /*Payoff control variate*/
  aux+=(Price[k*OP_Exercise_Dates+j+1]+
      (p->Compute) (p->Par,*(Mesh+k*OP_Exerc
  ise_Dates*BS_Dimension+(j+1)*BS_Dimension)))*Weig
  ht(j,Mesh+i*OP_Exercise_Dates*BS_Dimension+j*BS_
  Dimension,Mesh+k*OP_Exercise_Dates*BS_Dimension+(
  j+1)*BS_Dimension,k,BS_Dimension,Step);
  }
  aux*=DiscountStep/(double)AL_Mesh_Size;
  aux-=(p->Compute)(p->Par,*(Mesh+i*OP_Exercis
  e_Dates*BS_Dimension+j*BS_Dimension));
  Price[i*OP_Exercise_Dates+j]=MAX(0,aux);
}
}

aux=0;
for (i=0;i<AL_Mesh_Size;i++){
aux+=Price[i*OP_Exercise_Dates+1]+(p->Compute)
   (p->Par,*(Mesh+i*OP_Exercise_Dates*BS_Dimension
  +BS_Dimension));
}

/*Backward Price*/
if(!gj_flag)
AL_BPrice=MAX((p->Compute)(p->Par,*(BS_Spot)),
  DiscountStep*aux/(double)AL_Mesh_Size);
else
AL_BPrice=DiscountStep*aux/(double)AL_Mesh_Siz
  e;
```

```
/* Forward Price */
AL_FPrice=0;
for(i=0;i<AL_MonteCarlo_Iterations;i++){

for (l=0;l<BS_Dimension;l++){
  Path[l]=BS_Spot[l];
}

j=0;
do {
  aux=0;
  for (k=0;k<AL_Mesh_Size;k++){
  aux+=(Price[k*OP_Exercise_Dates+j+1]+(p->
  Compute)(p->Par,*(Mesh+k*OP_Exercise_Dates*BS_Dimen
  sion+(j+1)*BS_Dimension)))*Weight(j,Path+j*BS_Dim
  ension,Mesh+k*OP_Exercise_Dates*BS_Dimension+(j+1
  )*BS_Dimension,k,BS_Dimension,Step);
  }
  aux*=DiscountStep/(double)AL_Mesh_Size;
  aux-=(p->Compute)(p->Par,*(Path+j*BS_Dimensi
  on));

  j++;
  BS_Forward_Step(mc_or_qmc,generator,Path+j*
  BS_Dimension,Path+(j-1)*BS_Dimension,BS_Dimension,
  Step,Sqrt_Step);
}
while ((0<aux)&&(j<OP_Exercise_Dates-1));
AL_FPrice+=Discount((double)(j)*Step,BS_Inter
  est_Rate)*(p->Compute) (p->Par,*(Path+(j)*BS_Dim
  ension));
}
AL_FPrice/=(double)AL_MonteCarlo_Iterations;

/*Price = Mean of Forward and Backward Price*/
*AL_Price=0.5*(AL_FPrice+AL_BPrice);

/*Memory Disallocation*/
if (AL_ShuttingDown){
Brod_Liberation();
```

```
  }
}



static int MCBroadieGlassermann(double s,
    NumFunc_1  *p, double t, double r, double dividend,
    double sig, long N, int generator, double inc,int mes
    h_size, int exercise_date_number,double *ptprice,
     double *ptdelta)
{


  double p1,p2,p3;
  int simulation_dim= 1,fermeture=1,init_mc,
    mc_or_qmc;
  double s_vector[1];
  double s_vector_plus[1];
  double divid[1];
  double sigma[1];

  /*Initialisation*/
  s_vector[0]=s;
  s_vector_plus[0]=s*(1.+inc);
  divid[0]=dividend;
  sigma[0]=sig;
  /*MC sampling*/
  init_mc= InitGenerator(generator,simulation_dim
    ,N);

  /* Test after initialization for the generator
    */
  if(init_mc == OK)
    { mc_or_qmc= Rand_Or_Quasi(generator);

  /*Geske-Johnson Formulae*/
  if (exercise_date_number==0) {
    BrGl(&p3,N,p,mesh_size,fermeture,mc_or_qmc,g
    enerator,4,s_vector,t,r,divid,sigma,1);
    BrGl(&p2,N,p,mesh_size,fermeture,mc_or_qmc,g
    enerator,3,s_vector,t,r,divid,sigma,1);
```

```
    BrGl(&p1,N,p,mesh_size,fermeture,mc_or_qmc,g
    enerator,2,s_vector,t,r,divid,sigma,1);
    *ptprice=p3+7./2.*(p3-p2)-(p2-p1)/2.;
  } else {
    BrGl(ptprice,N,p,mesh_size,fermeture,mc_or_q
    mc,generator,exercise_date_number,s_vector,t,r,div
    id,sigma,0);
  }

  /*Delta*/
  if (exercise_date_number==0) {
    BrGl(&p3,N,p,mesh_size,fermeture,mc_or_qmc,g
    enerator,4,s_vector_plus,t,r, divid,sigma,1);
    BrGl(&p2,N,p,mesh_size,fermeture,mc_or_qmc,g
    enerator,3,s_vector_plus,t,r,divid,sigma,1);
    BrGl(&p1,N,p,mesh_size,fermeture,mc_or_qmc,g
    enerator,2,s_vector_plus,t,r,divid,sigma,1);
    *ptdelta=((p3+7./2.*(p3-p2)-(p2-p1)/2.)-*pt
    price)/(s*inc);
  } else {
    BrGl(&p1,N,p,mesh_size,fermeture,mc_or_qmc,g
    enerator,exercise_date_number,s_vector_plus,t,r,
    divid,sigma,0);
    *ptdelta=(p1-*ptprice)/(s*inc);
  }
  }
  return init_mc;
}

int CALC(MC_BroadieGlassermann)(void *Opt, void *
    Mod, PricingMethod *Met)
{
  TYPEOPT* ptOpt=(TYPEOPT*)Opt;
  TYPEMOD* ptMod=(TYPEMOD*)Mod;
  double r,divid;

  r=log(1.+ptMod->R.Val.V_DOUBLE/100.);
  divid=log(1.+ptMod->Divid.Val.V_DOUBLE/100.);

  return MCBroadieGlassermann(ptMod->S0.Val.V_PD
    OUBLE,
```

```
                ptOpt->PayOff.Val.V_
    NUMFUNC_1,
                ptOpt->Maturity.Val.V_DA
    TE-ptMod->T.Val.V_DATE,
                r,
                divid,
                ptMod->Sigma.Val.V_PDOUBL
    E,
                Met->Par[0].Val.V_LONG,
                Met->Par[1].Val.V_INT,
                Met->Par[2].Val.V_PDOUBLE,
                Met->Par[3].Val.V_INT,
                Met->Par[4].Val.V_INT,
                &(Met->Res[0].Val.V_
    DOUBLE),
                &(Met->Res[1].Val.V_
    DOUBLE));
}

int CHK_OPT(MC_BroadieGlassermann)(void *Opt, vo
    id *Mod)
{
  Option* ptOpt=(Option*)Opt;
  TYPEOPT* opt=(TYPEOPT*)(ptOpt->TypeOpt);

  if ((opt->EuOrAm).Val.V_BOOL==AMER)
    return OK;
else
    return WRONG;
}



static int MET(Init)(PricingMethod *Met)
{
  static int first=1;
  int type_generator;

  type_generator= Met->Par[1].Val.V_INT;

  if (first)
```

```
    {
      Met->Par[0].Val.V_LONG=10000;
      Met->Par[1].Val.V_INT=0;
      Met->Par[2].Val.V_PDOUBLE=0.1;
    Met->Par[3].Val.V_INT=200;
    Met->Par[4].Val.V_INT=10;
    first=0;
    }
  return OK;
}

PricingMethod MET(MC_BroadieGlassermann)=
{
  "MC_BroadieGlassermann",
  {{"N iterations",LONG,100,ALLOW},
   {"RandomGenerator",GENER,100,ALLOW},
   {"Delta Increment Rel",PDOUBLE,100,ALLOW},
   {"Mesh Size",INT,100,ALLOW},
   {"Number of Exercise Dates (0->Geske Johnson
    Formulae",INT,100,ALLOW},
   {" ",END,0,FORBID}},
  CALC(MC_BroadieGlassermann),
  {{"Price",DOUBLE,100,FORBID},
   {"Delta",DOUBLE,100,FORBID} ,
   {" ",END,0,FORBID}},
  CHK_OPT(MC_BroadieGlassermann),
  CHK_mc,
  MET(Init)
};
```

References