Help
```
#include <stdlib.h>
#include  "bs2d_std2d.h"
#include "error_msg.h"
#include "math/mc_am.h"
#include "enums.h"

static double *Mesh=NULL;
static long *Weights=NULL;
static double *Path=NULL, *Mean_Cell=NULL, *Price=NULL, *
    Transition=NULL, *PathAux=NULL;
static double *PathAuxPO=NULL,*Aux_BS=NULL,*Sigma=NULL,*Aux
    _Stock=NULL;

static int BaMa_Allocation(int AL_PO_Size,int BS_Dimension,
         int OP_Exercise_Dates)
{
  if (Mesh==NULL)
    Mesh= malloc(OP_Exercise_Dates*(AL_PO_Size+1)*sizeof(
    double));
  if (Mesh==NULL) return MEMORY_ALLOCATION_FAILURE;

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

  if (Weights==NULL)
    Weights= malloc(OP_Exercise_Dates*AL_PO_Size*sizeof(lon
    g));

  if (Weights==NULL)
    return MEMORY_ALLOCATION_FAILURE ;


  if (Mean_Cell==NULL)
    Mean_Cell= malloc(OP_Exercise_Dates*AL_PO_Size*sizeof(
    double));
  if (Mean_Cell==NULL) return MEMORY_ALLOCATION_FAILURE;
```

```
if (Price==NULL)
  Price= malloc(OP_Exercise_Dates*AL_PO_Size*sizeof(
  double));
if (Price==NULL)
  return MEMORY_ALLOCATION_FAILURE;

if (Transition==NULL)
  Transition= malloc((OP_Exercise_Dates-1)*AL_PO_Size*AL_
  PO_Size*sizeof(double));
if (Transition==NULL)
  return MEMORY_ALLOCATION_FAILURE;

if (PathAux==NULL)
  PathAux= malloc(AL_PO_Size*2*BS_Dimension*sizeof(
  double));
if (PathAux==NULL)
  return MEMORY_ALLOCATION_FAILURE;

if (PathAuxPO==NULL)
  PathAuxPO= malloc((AL_PO_Size+1)*sizeof(double));
if (PathAuxPO==NULL)
  return MEMORY_ALLOCATION_FAILURE;

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


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

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

```
  return OK;
}

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

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

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

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

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

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

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

  if (PathAuxPO!=NULL) {
```

```
    free(PathAuxPO);
    PathAuxPO=NULL;
  }

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

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

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

/*Black-Sholes Step*/
static void BS_Forward_Step(int generator,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*pnl_rand_normal(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*Aux_BS[j];
    Stock[j]=Initial_Stock[j]*exp(Aux);
  }
}
```

```
/*Cell Number in the mesh*/
static int Number_Cell(double x, int Instant, int AL_PO_Si
    ze)
{
  int min=0,max=AL_PO_Size,j;

  do {
    j=(max+min)/2;
    if (x>=Mesh[Instant*(AL_PO_Size+1)+j]) {
      min=j;
    } else {
      max=j;
    }
  } while (!((x>=Mesh[Instant*(AL_PO_Size+1)+j])&&(x<=Mesh[
    Instant*(AL_PO_Size+1)+j+1])));

  return j;
}

/*Black-Sholes Path*/
static void ForwardPath(double *Path, double *Initial_Stock
    , int Initial_Time,int Number_Dates,int generator,int BS_
    Dimension, double Step, double Sqrt_Step)
{
  int i,j,k;
  double aux;
  double *SigmapjmBS_Dimensionpk;

  for (j=0;j<BS_Dimension;j++) Path[Initial_Time*BS_Dimens
    ion+j]=Initial_Stock[j];

  for (i=Initial_Time+1;i<Initial_Time+Number_Dates;i++){
    for (j=0;j<BS_Dimension;j++){
      Aux_Stock[j]=Sqrt_Step*pnl_rand_normal(generator);
    }
    SigmapjmBS_Dimensionpk=Sigma;


    for (j=0;j<BS_Dimension;j++){
      aux=0.;
      for (k=0;k<=j;k++){
```

```
        aux+=(*SigmapjmBS_Dimensionpk)*Aux_Stock[k];
  SigmapjmBS_Dimensionpk++;
      }
      SigmapjmBS_Dimensionpk+=BS_Dimension-j-1;
      aux-=Step*Aux_BS[j];
      Path[i*BS_Dimension+j]=Path[(i-1)*BS_Dimension+j]*exp
    (aux);
    }
  }
}

static void Init_Cells(int generator,NumFunc_2 *p,int BS_
    Dimension, int OP_Exercise_Dates, int AL_MonteCarlo_Iteratio
    ns,int AL_PO_Size, double *BS_Spot,double Step, double Sq
    rt_Step)
{
  double auxop1,auxop2;
  int i,j,k,auxcell1,auxcell2;

  for (i=0;i<OP_Exercise_Dates-1;i++)
    for (j=0;j<AL_PO_Size;j++)
      for (k=0;k<AL_PO_Size;k++) Transition[i*AL_PO_Size*
    AL_PO_Size+j*AL_PO_Size+k]=0;
  for (i=0;i<OP_Exercise_Dates;i++)
    for (j=0;j<AL_PO_Size;j++) Mean_Cell[i*AL_PO_Size+j]=0;
  for (i=0;i<OP_Exercise_Dates;i++)
    for (j=0;j<AL_PO_Size;j++) Price[i*AL_PO_Size+j]=0;
  for (i=0;i<OP_Exercise_Dates;i++)
    for (j=0;j<AL_PO_Size;j++) Weights[i*AL_PO_Size+j]=0;

  for (k=0;k<AL_MonteCarlo_Iterations;k++){

    ForwardPath(Path,BS_Spot,0,OP_Exercise_Dates,    generator,BS_Dimension,Step

    auxop2=(p->Compute)(p->Par,*Path,*(Path+1));
    auxcell2=Number_Cell(auxop2,0,AL_PO_Size);

    for (i=0;i<OP_Exercise_Dates-1;i++){
      auxcell1=auxcell2;
      auxop1=auxop2;
      auxop2=(p->Compute)(p->Par,*(Path+(i+1)*BS_Dimension)
```

```
    ,*(Path+(i+1)*BS_Dimension+1));
      auxcell2=Number_Cell(auxop2,i+1,AL_PO_Size);
      Weights[i*AL_PO_Size+auxcell1]++;
      Transition[i*AL_PO_Size*AL_PO_Size+auxcell1*AL_PO_Si
    ze+auxcell2]++;
      Mean_Cell[i*AL_PO_Size+auxcell1]+=auxop1;
    }
    auxop1=(p->Compute) (p->Par,*(Path+(OP_Exercise_Dates-1
    )*BS_Dimension),
       *(Path+(OP_Exercise_Dates-1)*BS_Dimension+1));
    auxcell1=Number_Cell(auxop1,OP_Exercise_Dates-1,AL_PO_
    Size);
    Weights[(OP_Exercise_Dates-1)*AL_PO_Size+auxcell1]++;
    Mean_Cell[(OP_Exercise_Dates-1)*AL_PO_Size+auxcell1]+=
    auxop1;
  }
}

static void InitMesh(NumFunc_2 *p,int generator,int AL_PO_
    Size, long Al_PO_Init, int BS_Dimension,int OP_Exercise_Da
    tes,double *BS_Spot, double Step, double Sqrt_Step)
{
  int i,j,k,l;

  for (i=0;i<OP_Exercise_Dates*(AL_PO_Size+1);i++)
    Mesh[i]=0;

  for (i=0;i<Al_PO_Init;i++){

    for (j=0;j<AL_PO_Size;j++){
      for (k=0;k<BS_Dimension;k++){
  PathAux[j*2*BS_Dimension+k]=BS_Spot[k];
      }
    }

    for (j=1;j<OP_Exercise_Dates;j++){
      for (k=0;k<AL_PO_Size;k++){
  BS_Forward_Step(generator,PathAux+k*2*BS_Dimension+BS_
    Dimension,PathAux+k*2*BS_Dimension,BS_Dimension,Step,Sqrt_
    Step);
      }
```

```
      for (k=1;k<AL_PO_Size+1;k++){
  PathAuxPO[k]= (p->Compute)(p->Par,*(PathAux+(k-1)*2*BS_
    Dimension+BS_Dimension),*(PathAux+(k-1)*2*BS_Dimension+BS_
    Dimension+1));
      }

      Sort(AL_PO_Size,PathAuxPO);

      for (k=1;k<AL_PO_Size+1;k++){
  Mesh[j*(AL_PO_Size+1)+k]+=PathAuxPO[k];
      }

      for (l=0;l<AL_PO_Size;l++){
  for (k=0;k<BS_Dimension;k++){
    PathAux[l*2*BS_Dimension+k]=PathAux[(l*2+1)*BS_Dimens
    ion+k];
  }
      }
    }
  }

  for (j=1;j<OP_Exercise_Dates;j++){
    for (k=1;k<AL_PO_Size+1;k++){
      Mesh[j*(AL_PO_Size+1)+k]/=(double)Al_PO_Init;
    }
  }

  for (j=1;j<OP_Exercise_Dates;j++){
    Mesh[j*(AL_PO_Size+1)]=0;
    Mesh[(j+1)*(AL_PO_Size+1)-1]=DBL_MAX;
    for (k=1;k<AL_PO_Size-1;k++)
      Mesh[j*(AL_PO_Size+1)+k]=(Mesh[j*(AL_PO_Size+1)+k]+
        Mesh[j*(AL_PO_Size+1)+k+1])*0.5;
  }
  Mesh[AL_PO_Size]=DBL_MAX;
  for (k=0;k<AL_PO_Size;k++)
    Mesh[k]=0;
}
```

```
static void BaMa(double *AL_BPrice,long AL_MonteCarlo_Itera
    tions,NumFunc_2 *p,int AL_PO_Size,int AL_PO_Init,int AL_Shut
    tingDown,int generator,int OP_Exercise_Dates,double *BS_Spo
    t, double BS_Maturity, double BS_Interest_Rate, double *div
    id, double *sigma,int gj_flag)
{
  double aux,Step,Sqrt_Step,DiscountStep;
  int i,j,k;
  int BS_Dimension;

  BS_Dimension=2;

  /*Memory Allocation*/
  BaMa_Allocation(AL_PO_Size,BS_Dimension,OP_Exercise_Da
    tes);
  *AL_BPrice=0.;
  Step=BS_Maturity/(double)(OP_Exercise_Dates-1);
  Sqrt_Step=sqrt(Step);
  DiscountStep=exp(-BS_Interest_Rate*Step);

  /*Black-Sholes initalization parameters*/
  Sigma[0]=sigma[0];
  Sigma[1]=sigma[1];
  Sigma[2]=sigma[2];
  Sigma[3]=sigma[3];

  Aux_BS[0]=0.5*(SQR(sigma[0])+SQR(sigma[1]))-BS_Interest_
    Rate+divid[0];
  Aux_BS[1]=0.5*(SQR(sigma[2])+SQR(sigma[3]))-BS_Interest_
    Rate+divid[1];

  /* Cells Weights and Transitions probabilities for the
    payoff mesh  */
  InitMesh(p,generator,AL_PO_Size,AL_PO_Init,BS_Dimension,
    OP_Exercise_Dates,BS_Spot,Step,Sqrt_Step);
  Init_Cells(generator,p,BS_Dimension,OP_Exercise_Dates,AL_
    MonteCarlo_Iterations,AL_PO_Size,BS_Spot,Step,Sqrt_Step);

  /*Initialization of the price at maturity*/
  for (k=0;k<AL_PO_Size;k++){
    if (Weights[(OP_Exercise_Dates-1)*AL_PO_Size+k]>0){
```

```
      Price[(OP_Exercise_Dates-1)*AL_PO_Size+k]=Mean_Cell[(
    OP_Exercise_Dates-1)*AL_PO_Size+k]/(double)Weights[(OP_Exerc
    ise_Dates-1)*AL_PO_Size+k];
    }
  }

  /* Dynamical programing (backward price)*/
  for (i=OP_Exercise_Dates-2;i>=0;i--){
    for (k=0;k<AL_PO_Size;k++){
      if (Weights[i*AL_PO_Size+k]>0){
      aux=0;
      for (j=0;j<AL_PO_Size;j++)
        aux+=Transition[i*AL_PO_Size*AL_PO_Size+k*AL_PO_Si
    ze+j]*Price[(i+1)*AL_PO_Size+j];
  aux/=(double)Weights[i*AL_PO_Size+k];
      aux*=DiscountStep;
  if((!gj_flag) || ((gj_flag)&&(i>0)))
    Price[i*AL_PO_Size+k]=MAX(Mean_Cell[i*AL_PO_Size+k]/We
    ights[i*AL_PO_Size+k],aux);
  else
    Price[k]=aux;
      }
    }
  }

  /*Backward Price*/
  *AL_BPrice=Price[Number_Cell((p->Compute)(p->Par,*BS_Spo
    t,*(BS_Spot+1)),0,AL_PO_Size)];

  /*Memory Disallocation*/
  if (AL_ShuttingDown){
    BaMa_Liberation();
  }
}

static int MCBarraquandMartineau2D(double s1, double s2,
    NumFunc_2  *p, double t, double r, double divid1, double divid2,
    double sigma1, double sigma2, double rho, long N, int     generator,  double
    double *ptprice, double *ptdelta1, double *ptdelta2)
{
```

```
double p1,p2,p3;
int simulation_dim= 1,fermeture=1,init_mc;
double s_vector[2];
double s_vector_plus1[2],s_vector_plus2[2];
double sigma[4];
double divid[2];


/* Covariance Matrix */
/* Coefficients of the matrix A such that A(tA)=Gamma */
sigma[0]= sigma1;
sigma[1]= 0.0;
sigma[2]= rho*sigma2;
sigma[3]= sigma2*sqrt(1.0-SQR(rho));

/*Initialisation*/
s_vector[0]=s1;
s_vector[1]=s2;
s_vector_plus1[0]=s1*(1.+inc);
s_vector_plus1[1]=s2;
s_vector_plus2[0]=s1;
s_vector_plus2[1]=s2*(1.+inc);
divid[0]=divid1;
divid[1]=divid2;

/*MC sampling*/
init_mc= pnl_rand_init(generator,simulation_dim,N);

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

    /*Geske-Johnson Formulae*/
    if (exercise_date_number==0) {
BaMa(&p3,N,p,size,init,fermeture,generator,4,s_vector,t,
  r,divid,sigma,1);
BaMa(&p2,N,p,size,init,fermeture,generator,3,s_vector,t,
  r,divid,sigma,1);
BaMa(&p1,N,p,size,init,fermeture,generator,2,s_vector,t,
  r,divid,sigma,1);
*ptprice=p3+7./2.*(p3-p2)-(p2-p1)/2.;
```

```
    } else {
  BaMa(ptprice,N,p,size,init,fermeture,generator,exercise_
    date_number,s_vector,t,r,divid,sigma,0);
    }

    /*Delta*/
    if (exercise_date_number==0) {
  BaMa(&p1,N,p,size,init,fermeture,generator,4,s_vector_pl
    us1,t,r,divid,sigma,1);
  BaMa(&p2,N,p,size,init,fermeture,generator,3,s_vector_pl
    us1,t,r,divid,sigma,1);
  BaMa(&p3,N,p,size,init,fermeture,generator,2,s_vector_pl
    us1,t,r,divid,sigma,1);
  *ptdelta1=((p3+7./2.*(p3-p2)-(p2-p1)/2.)-*ptprice)/(s1*
    inc);

  BaMa(&p1,N,p,size,init,fermeture,generator,4,s_vector_pl
    us2,t,r,divid,sigma,1);
  BaMa(&p2,N,p,size,init,fermeture,generator,3,s_vector_pl
    us2,t,r,divid,sigma,1);
  BaMa(&p3,N,p,size,init,fermeture,generator,2,s_vector_pl
    us2,t,r,divid,sigma,1);
  *ptdelta2=((p3+7./2.*(p3-p2)-(p2-p1)/2.)-*ptprice)/(s2*
    inc);

    } else {
  BaMa(&p1,N,p,size,init,fermeture,generator,exercise_da
    te_number,s_vector_plus1,t,r,divid,sigma,0);
  *ptdelta1=(p1-*ptprice)/(s1*inc);
  BaMa(&p2,N,p,size,init,fermeture,generator,exercise_da
    te_number,s_vector_plus2,t,r,divid,sigma,0);
  *ptdelta2=(p2-*ptprice)/(s2*inc);
    }
  }
  return init_mc;
}

int CALC(MC_BarraquandMartineau2D)(void *Opt, void *Mod,
    PricingMethod *Met)
{
  TYPEOPT* ptOpt=(TYPEOPT*)Opt;
```

```
  TYPEMOD* ptMod=(TYPEMOD*)Mod;
  double r,divid1,divid2;

  r=log(1.+ptMod->R.Val.V_DOUBLE/100.);
  divid1=log(1.+ptMod->Divid1.Val.V_DOUBLE/100.);
  divid2=log(1.+ptMod->Divid1.Val.V_DOUBLE/100.);

  return MCBarraquandMartineau2D(ptMod->S01.Val.V_PDOUBLE,
         ptMod->S02.Val.V_PDOUBLE,
         ptOpt->PayOff.Val.V_NUMFUNC_2,
         ptOpt->Maturity.Val.V_DATE-ptMod->T.Val.V_DA
    TE,
         r,
         divid1,
         divid2,
         ptMod->Sigma1.Val.V_PDOUBLE,
         ptMod->Sigma2.Val.V_PDOUBLE,
         ptMod->Rho.Val.V_RGDOUBLE,
         Met->Par[0].Val.V_LONG,
         Met->Par[1].Val.V_ENUM.value,
         Met->Par[2].Val.V_PDOUBLE,
         Met->Par[3].Val.V_INT,
         Met->Par[4].Val.V_INT,
         Met->Par[5].Val.V_INT,
         &(Met->Res[0].Val.V_DOUBLE),
         &(Met->Res[1].Val.V_DOUBLE),&(Met->Res[2].Val.
    V_DOUBLE));
}

static int CHK_OPT(MC_BarraquandMartineau2D)(void *Opt, voi
    d *Mod)
{
  Option* ptOpt=(Option*)Opt;
  TYPEOPT* opt=(TYPEOPT*)(ptOpt->TypeOpt);

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

```
static int MET(Init)(PricingMethod *Met,Option *Opt)
{
  if ( Met->init == 0)
    {
      Met->init=1;

      Met->Par[0].Val.V_LONG=20000;
      Met->Par[1].Val.V_ENUM.value=0;
      Met->Par[1].Val.V_ENUM.members=&PremiaEnumMCRNGs;
      Met->Par[2].Val.V_PDOUBLE=0.1;
      Met->Par[3].Val.V_INT=100;
      Met->Par[4].Val.V_INT=300;
      Met->Par[5].Val.V_INT=10;

    }



  return OK;
}


PricingMethod MET(MC_BarraquandMartineau2D)=
{
  "MC_BarraquandMartineau2d",
  {{"N iterations",LONG,{100},ALLOW},
   {"RandomGenerator",ENUM,{100},ALLOW},
   {"Delta Increment Rel",PDOUBLE,{100},ALLOW},
   {"Number of Cells",INT,{100},ALLOW},
   {"Size of grid initialising sample",INT,{100},ALLOW},
   {"Number of Exercise Dates (0->Geske Johnson Formulae)",
    INT,{100},ALLOW},
   {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CALC(MC_BarraquandMartineau2D),
  {{"Price",DOUBLE,{100},FORBID},
   {"Delta1",DOUBLE,{100},FORBID} ,{"Delta2",DOUBLE,{100},
    FORBID},
   {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CHK_OPT(MC_BarraquandMartineau2D),
  CHK_mc,
```

```
  MET(Init)
};
```

References