```
   Help
#include  "bs2d_std2d.h"
#include "error_msg.h"

static int KamradRitchken_91(int am,double s1,double s2,
    NumFunc_2  *p,double t,double r,double divid1,double divid2,
    double sigma1,double sigma2,double rho,int N,double lambda,
                              double *ptprice,double *ptdelt
    a1,double *ptdelta2)
{
  int npoints,i,j,k;
  double h,m1,m2,u1,u2,d1,d2,lowerstock1,lowerstock2,puu,
    pud,pdu,pdd,pm,stock1,stock2;
  double **iv,**P;

  /*Memory Allocation*/
  iv=(double **)calloc(2*N+1,sizeof(double *));
  if (iv==NULL)
    return MEMORY_ALLOCATION_FAILURE;
  for (i=0;i<2*N+1;i++)
    {
      iv[i]=(double *)calloc(2*N+1,sizeof(double));
      if (iv[i]==NULL)
  return MEMORY_ALLOCATION_FAILURE;
    }

  P=(double **)calloc(2*N+1,sizeof(double *));
  if (P==NULL)
    return MEMORY_ALLOCATION_FAILURE;
  for (i=0;i<2*N+1;i++)
    {
      P[i]=(double *)calloc(2*N+1,sizeof(double));
      if (P[i]==NULL)
  return MEMORY_ALLOCATION_FAILURE;
    }

  /*Up and Down factors*/
  h=t/(double)N;
  u1=exp(lambda*sigma1*sqrt(h));
  d1= 1.0/u1;
  u2=exp(lambda*sigma2*sqrt(h));
```

```
d2= 1.0/u2;

/*Risk-Neutral Probabilities*/
m1=(r-divid1)-SQR(sigma1)/2.0;
m2=(r-divid2)-SQR(sigma2)/2.0;

puu=exp(-r*h)*(1./SQR(lambda)+(sqrt(h)/lambda)*(m1/sigma1
  +m2/sigma2)+rho/SQR(lambda))/4.;
pud=exp(-r*h)*(1./SQR(lambda)+(sqrt(h)/lambda)*(m1/sigma1
  -m2/sigma2)-rho/SQR(lambda))/4.;
pdu=exp(-r*h)*(1./SQR(lambda)+(sqrt(h)/lambda)*(-m1/sigma
  1+m2/sigma2)-rho/SQR(lambda))/4.;
pdd=exp(-r*h)*(1./SQR(lambda)+(sqrt(h)/lambda)*(-m1/sigma
  1-m2/sigma2)+rho/SQR(lambda))/4.;
pm=exp(-r*h)*(1.-1./SQR(lambda));

/*Terminal Values*/
lowerstock1=s1;lowerstock2=s2;
for(i=0;i<N;i++)
  {
    lowerstock1*=d1;
    lowerstock2*=d2;
  }

stock1=lowerstock1;stock2=lowerstock2;
for (i=0;i<=2*N;i++,stock1*=u1,stock2=lowerstock2)
  for (j=0;j<=2*N;j++,stock2*=u2)
    {
iv[i][j]=(p->Compute)(p->Par,stock1,stock2);
P[i][j]=iv[i][j];
    }

/*Backward Cycle*/
npoints=2*N;
for (k=1;k<=N-1;k++)
  {
    npoints-=2;
    for(i=0;i<=npoints;i++)
for(j=0;j<=npoints;j++)
  {
    P[i][j]=pdu*P[i][j+2]+puu*P[i+2][j+2]+pdd*P[i][j]+
```

```
    pud*P[i+2][j]+pm*P[i+1][j+1];
      if (am)
        P[i][j]= MAX(iv[k+i][k+j],P[i][j]);
    }

    }
  /*Deltas*/
  MOD_OPT(Delta_Operator)(u1,d1,u2,d2,s1,s2,P[2][2],P[2][0]
    ,P[0][2],P[0][0],ptdelta1,ptdelta2);

  /*First Time Step*/
  P[0][0]=pdu*P[0][2]+puu*P[2][2]+pdd*P[0][0]+pud*P[2][0]+
    pm*P[1][1];
  if (am)
    P[0][0]=MAX(iv[N][N],P[0][0]);

  /*Price*/
  *ptprice=P[0][0];

  /*Memory desallocation*/
  for (i=0;i<2*N+1;i++)
    free(iv[i]);
  free(iv);

  for (i=0;i<2*N+1;i++)
    free(P[i]);
  free(P);

  return OK;
}

int CALC(TR_KamradRitchken)(void *Opt,void *Mod,Pricing
    Method *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->Divid2.Val.V_DOUBLE/100.);
```

```
  return KamradRitchken_91(ptOpt->EuOrAm.Val.V_BOOL,
         ptMod->S01.Val.V_PDOUBLE,ptMod->S02.Val.V_PDOUB
    LE,
         ptOpt->PayOff.Val.V_NUMFUNC_2,ptOpt->Maturity.
    Val.V_DATE-ptMod->T.Val.V_DATE,r,divid1,divid2,
         ptMod->Sigma1.Val.V_PDOUBLE,ptMod->Sigma2.Val.
    V_PDOUBLE,ptMod->Rho.Val.V_RGDOUBLE,
         Met->Par[0].Val.V_INT,Met->Par[1].Val.V_RG
    DOUBLE,
         &(Met->Res[0].Val.V_DOUBLE),&(Met->Res[1].Val.
    V_DOUBLE),&(Met->Res[2].Val.V_DOUBLE));
}

static int CHK_OPT(TR_KamradRitchken)(void *Opt, void *Mod)
{
  return  OK;
}

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

      Met->Par[0].Val.V_INT2=100;
      Met->Par[1].Val.V_RGDOUBLE12=1.22;


    }

  return OK;
}

PricingMethod MET(TR_KamradRitchken)=
{
  "TR_KamradRitchken",
  {{"StepNumber",INT2,{100},ALLOW},{"Lambda",RGDOUBLE12,{10
    0},ALLOW},{" ",PREMIA_NULLTYPE,{0},FORBID}},
  CALC(TR_KamradRitchken),
  {{"Price",DOUBLE,{100},FORBID},{"Delta1",DOUBLE,{100},FO
```

```
    RBID} ,{"Delta2",DOUBLE,{100},FORBID} ,
   {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CHK_OPT(TR_KamradRitchken),
  CHK_tree,
  MET(Init)
};
```

References