```
   Help
#include  "bergomirev2d_vol.h"

#include <pnl/pnl_mathtools.h>
#include <pnl/pnl_list.h>
#include <pnl/pnl_integration.h>
#include <pnl/pnl_cdf.h>
#include <pnl/pnl_random.h>
#include <pnl/pnl_finance.h>
#include <pnl/pnl_vector_double.h>
#include <pnl/pnl_basis.h>

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2012+2) //The "#else" part of the code will be freely av
    ailable after the (year of creation of this file + 2)
static int CHK_OPT(AP_EXPANSION_OA_BERGOMIREV)(void *Opt,
    void *Mod)
{
  return NONACTIVE;
}
int CALC(AP_EXPANSION_OA_BERGOMIREV)(void*Opt,void *Mod,
    PricingMethod *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else

static PnlMat *ResBergomi;
typedef struct params_bergomi {
  int n;
  double VarianceCurve;
  double Gamma;
  double Omega1;
  double Omega2;
  double Correl;


  double Small;
  double VIXFuture;
  double PutVIX;
  double Strike;
```

```
} params_bergomi;


static double g(double x, params_bergomi *p)
{
  double m_dVarianceCurve;
  double m_dGamma ;
  double m_dOmega1 ;
  double m_dOmega2 ;

  m_dVarianceCurve= p->VarianceCurve;
  m_dGamma =  p->Gamma;
  m_dOmega1 = p->Omega1;
  m_dOmega2 = p->Omega2;

  return (m_dVarianceCurve * ((1.0 - m_dGamma) * exp(m_dOm
    ega1 * x - m_dOmega1 * m_dOmega1 / 2.0) + m_dGamma * exp(m_
    dOmega2 * x - m_dOmega2 * m_dOmega2 / 2.0)));
}


static double z(double x, params_bergomi *p)
{
  double m_dVarianceCurve;
  double m_dGamma ;
  double m_dOmega1 ;
  double m_dOmega2 ;
  double error, leght,middle;
  double LowerBond, UperBond;

  m_dVarianceCurve= p->VarianceCurve;
  m_dGamma =  p->Gamma;
  m_dOmega1 = p->Omega1;
  m_dOmega2 = p->Omega2;

  if (m_dOmega1 == 0)
    return 0;

  if (m_dOmega2 == 0)
    {
```

```
    if (m_dVarianceCurve * m_dGamma == x)
return (m_dOmega1 / 2.0);
    else
return (m_dOmega1 / 2.0 + log((x / m_dVarianceCurve - m_
  dGamma) / (1 - m_dGamma)) / m_dOmega1);

  }
if (x <= g(0,p))
  {
    UperBond = 0;
    LowerBond = -1.0;
    while (x <= g(LowerBond,p))
{
  UperBond = LowerBond;
  LowerBond -= 1.0;
}
  }
else
  {
    UperBond = 1;
    LowerBond = 0.0;
    while (x >= g(UperBond,p))
{
  LowerBond = UperBond;
  UperBond += 1.0;
}
  }
error = 1;
leght = 1;
middle = (LowerBond + UperBond) / 2.0;
while (error >= p->Small && leght >= p->Small)
  {
    middle = (LowerBond + UperBond) / 2.0;
    if (g(middle,p) < x)
LowerBond = middle;
    else
UperBond = middle;
    error = ABS(x - g(middle,p));
    leght = UperBond - LowerBond;
  }
return middle;
```

```
}



static double call_by_density(double k, params_bergomi *p)
{
  int path,i ;
  double sup ;
  PnlVect *axis;
  double strikeMin, res;

    path = 2000;
  sup = 12;
  axis = pnl_vect_create(path);

  strikeMin = z(k * k,p);


  res = 0;

  for (i = 0; i < path; i++)
    pnl_vect_set(axis,i,strikeMin + sup * i / (double)path)
    ;
  for (i = 0; i < path - 1; i++)
    res += (pnl_vect_get(axis,i+1) - pnl_vect_get(axis,i))
    * MAX(sqrt(g(pnl_vect_get(axis,i),p)) - k, 0.0) *
      exp(-pnl_vect_get(axis,i) * pnl_vect_get(axis,i) / 2.
    0) / sqrt(2.0 * M_PI);

  pnl_vect_free(&axis);
  return res;
}

//Put price under Bergomi Revisited
static double put_by_density(double k, params_bergomi *p)
{
  int path,i ;
  double inf ;
  PnlVect *axis;
  double strikeMax, res;
  path = 2000;
  inf = -12;
  axis = pnl_vect_create(path);

  strikeMax = z(k * k,p);


  res = 0;

  for (i = 0; i < path; i++)
    pnl_vect_set(axis,i,inf + (i + 1) * (strikeMax - inf) /
     (double)path);
  for (i = 0; i < path - 1; i++)
    res += (pnl_vect_get(axis,i+1) - pnl_vect_get(axis,i))
    * MAX(k - sqrt(g(pnl_vect_get(axis,i),p)), 0.0) *
      exp(-pnl_vect_get(axis,i) * pnl_vect_get(axis,i) / 2.
    0) / sqrt(2.0 * M_PI);

    pnl_vect_free(&axis);

  return res;
}


static void RowFromFile(char *chaine, int numCol, PnlVect *
    res)
{
  int i=0;
  char delims[] = "{t";
  char *result = NULL;
  result = strtok( chaine, delims );
  while( result != NULL ) {
    pnl_vect_set(res,i, atof ( result ));

    result = strtok( NULL, delims );
    i++;
  }

}
```

```
static PnlMat *RedFilMatrix(FILE* FVParams  )
{
  PnlVect *aux;
  char chaine[1000] = "";
  int NumberMat=0,i,j;
  int TAILLE_MAX=1000;

  if(FVParams != NULL)
    {
      if(fgets(chaine, TAILLE_MAX, FVParams) != NULL)
  NumberMat = (int) atof ( chaine );

      ResBergomi = pnl_mat_create(NumberMat+1,3);

      aux = pnl_vect_create(3);
      pnl_mat_set(ResBergomi,0,0,(double) NumberMat);


      for(j=1;j<NumberMat+1;j++)
  {
    if(fgets(chaine, TAILLE_MAX, FVParams) != NULL)
      {
        RowFromFile(chaine, NumberMat, aux);

        for(i=0;i<3;i++)
    {
      pnl_mat_set(ResBergomi,j,i,pnl_vect_get(aux,i));

    }
      }
  }

      pnl_vect_free(&aux);
      fclose(FVParams);
    }
  return ResBergomi;
}


static int getIndex(PnlMat *FVParams, double T)
```

```
{

  int i = 1;
  if(pnl_mat_get(FVParams,1,0) > T)
    return 1;

  while(i<= (int)pnl_mat_get(FVParams,0,0) && pnl_mat_get(
    FVParams,i,0) <= T )
    i++;


  return i-1;
}

static void PutVIXBergomiRev(double k1 , double k2, double
    Theta, double RhoXY, double T, double K, double *price, FIL
    E* fvParams)
{

  params_bergomi p;
  double zeta;
  double beta;
  double gamma;
  double KHI0;
  PnlMat *ForVar;
  int Index;

  ForVar = RedFilMatrix(fvParams );

  Index = getIndex(ForVar, T);

  if(Index < (int)pnl_mat_get(ForVar,0,0)-1)
    {
      gamma = ( pnl_mat_get(ForVar,Index+2,0) - pnl_mat_get
    (ForVar,Index+1,0))/( pnl_mat_get(ForVar,Index+2,0) - pnl_
    mat_get(ForVar,Index,0));
      beta = pnl_mat_get(ForVar,Index+1,1)/pnl_mat_get(ForV
    ar,Index,1);
    }
  else
    {
      gamma = 0.0;
      beta = 0.0;
    }


  p.n = 0;

  zeta = pnl_mat_get(ForVar,Index,1);
  KHI0 = pnl_mat_get(ForVar,Index,2);
  p.Correl = 0;

  p.Small = 0.000001;
  p.Omega1 = zeta;
  p.Omega2 = beta*zeta;
  p.Gamma = gamma;
  p.VarianceCurve = KHI0;

  pnl_mat_free(&ResBergomi);

  *price = put_by_density(K,&p);

}

static void CallVIXBergomiRev(double k1 , double k2,
    double Theta, double RhoXY, double T, double K, double *price,
    FILE* fvParams)
{

  params_bergomi p;
  double zeta;
  double beta;
  double gamma;
  double KHI0;
  PnlMat *ForVar;
  int Index;

  ForVar = RedFilMatrix(fvParams );

  Index = getIndex(ForVar, T);

  if(Index < (int)pnl_mat_get(ForVar,0,0)-1)
```

```
    {
      gamma = ( pnl_mat_get(ForVar,Index+2,0) - pnl_mat_get
    (ForVar,Index+1,0))/( pnl_mat_get(ForVar,Index+2,0) - pnl_
    mat_get(ForVar,Index,0));
      beta = pnl_mat_get(ForVar,Index+1,1)/pnl_mat_get(ForV
    ar,Index,1);
    }
  else
    {
      gamma = 0.0;
      beta = 0.0;
    }


  p.n = 0;

  zeta = pnl_mat_get(ForVar,Index,1);
  KHI0 = pnl_mat_get(ForVar,Index,2);
  p.Correl = 0;

  p.Small = 0.000001;
  p.Omega1 = zeta;
  p.Omega2 = beta*zeta;
  p.Gamma = gamma;
  p.VarianceCurve = KHI0;

  pnl_mat_free(&ResBergomi);

  *price = call_by_density(K,&p);

}
int ApOptionVIXBergomiRev(double S0,NumFunc_1  *p, double
    t,double r,char *ForwardVarianceData,double Theta, double
    k1,double k2,double *ptprice)
{
  int flag_call;
  double RhoXY=-0. ;
  double K,price;
  FILE* FVPARAMS= NULL;
  FVPARAMS = fopen(ForwardVarianceData, "r");
  if ((p->Compute)==&Call)
    flag_call=1;
  else
    flag_call=0;

  K=p->Par[0].Val.V_PDOUBLE;

  //Put Case
  if(flag_call==0)
    {
      PutVIXBergomiRev( k1 , k2, Theta, RhoXY,   t,  K, &
    price,FVPARAMS);
      *ptprice=price*exp(-r*t);
    }
  else//Call Case
    {
      CallVIXBergomiRev( k1 , k2, Theta, RhoXY,   t,  K, &
    price,FVPARAMS);
      *ptprice=price*exp(-r*t);
    }

  return OK;
}

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

  r=log(1.+ptMod->R.Val.V_DOUBLE/100.);
  return  ApOptionVIXBergomiRev(ptMod->S0.Val.V_PDOUBLE,
        ptOpt->PayOff.Val.V_NUMFUNC_1,
        ptOpt->Maturity.Val.V_DATE-ptMod->T.Val.V_DATE,
                                r,
                                ptMod->ForwardVarianceData.
    Val.V_FILENAME,
        ptMod->theta.Val.V_PDOUBLE
        ,ptMod->k1.Val.V_PDOUBLE,
        ptMod->k2.Val.V_PDOUBLE,
        //ptMod->rhoxy.Val.V_RGDOUBLE,
        &(Met->Res[0].Val.V_DOUBLE));
}

static int CHK_OPT(AP_OPTIONVIX_BERGOMIREV)(void *Opt, voi
    d *Mod)
{
  if ((strcmp(((Option*)Opt)->Name,"CallVixEuro")==0)||(
    strcmp( ((Option*)Opt)->Name,"PutVixEuro")==0))
    return OK;
  return WRONG;
}

#endif //PremiaCurrentVersion
static int MET(Init)(PricingMethod *Met,Option *Opt)
{
  if ( Met->init == 0)
    {
      Met->HelpFilenameHint = "AP_OULDALY";
      Met->init=1;
    }

  return OK;
}

PricingMethod MET(AP_OPTIONVIX_BERGOMIREV)=
{
  "AP_OULDALY",
  {{" ",PREMIA_NULLTYPE,{0},FORBID}},
  CALC(AP_OPTIONVIX_BERGOMIREV),
  {{"Price",DOUBLE,{100},FORBID} ,
   {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CHK_OPT(AP_OPTIONVIX_BERGOMIREV),
  CHK_ok,
  MET(Init)
};
```

References