```
   Help
#include <stdlib.h>
#include  "variancegamma1d_lim.h"
#include "pnl/pnl_vector_double.h"
#include "pnl/pnl_fft.h"
#include "pnl/pnl_cdf.h"
#include "math/wienerhopf.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2011+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_backwardfourierbar_vg)(void *Opt, voi
    d *Mod)
{
  return NONACTIVE;
}
int CALC(AP_backwardfourierbar_vg)(void*Opt,void *Mod,Prici
    ngMethod *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else

static int backwardfourier_vg_downout(int am, int upordown,
     int ifCall,
                         double Spot, double sigma,
                         double theta,double kappa,
                         double r, double divid,
                         double T, double h, double Strike1
    ,
                         double bar,double rebate,
                         double er, long int step,
                         double *ptprice, double *ptdelta)
{
  double  ptprice1, ptdelta1, mu, qu, om;
  double lm1, lp1, num, nup, cm, cp;

  double alfa, beta;
  double sig2=sigma*sigma;

  alfa=sqrt(theta*theta+2.0*sig2/kappa)/sig2;
```

```
beta=theta/sig2;
cp=1.0/kappa;
cm=cp;
lp1=alfa+beta;
lm1=beta - alfa;

if (upordown==0)
  {
    om=lm1<-2. ? 2. : (-lm1+1.)/2.;
  }
else
  {
    om= lp1>1. ? -1. : -lp1/2.;
  }

mu=r-divid+cp*(log(alfa*alfa-(beta+1)*(beta+1)) - log(
  alfa*alfa-beta*beta));

if (mu<0.0)
  {
    nup=1; num=0;
  }
else
  {
    nup=0; num=1;
  }

qu = r + cp*(log(alfa*alfa-(beta+om)*(beta+om)) - log(
  alfa*alfa-beta*beta)) - mu*om;

bi_barr(mu, qu, om, upordown, ifCall, Spot, lm1, lp1,
               num, nup, cm, cp, r, divid,
               T, h, Strike1, bar, rebate,
               er, step, &ptprice1, &ptdelta1);

//Price
*ptprice = ptprice1;
//Delta
*ptdelta = ptdelta1;

return OK;
```

```
}

//=========================================================
    ==========================
int CALC(AP_backwardfourierbar_vg)(void *Opt,void *Mod,
    PricingMethod *Met)
{
  TYPEOPT* ptOpt=( TYPEOPT*)Opt;
  TYPEMOD* ptMod=( TYPEMOD*)Mod;
  double r,divid,limit, strike, spot,rebate;

  NumFunc_1 *p;
  int res;
  int upordown;
  int ifCall;

  r=log(1.+ptMod->R.Val.V_DOUBLE/100.);
  divid=log(1.+ptMod->Divid.Val.V_DOUBLE/100.);
  limit=((ptOpt->Limit.Val.V_NUMFUNC_1)->Compute)((ptOpt->    Limit.Val.V_NUMFUN
  p=ptOpt->PayOff.Val.V_NUMFUNC_1;
  strike=p->Par[0].Val.V_DOUBLE;
  spot=ptMod->S0.Val.V_DOUBLE;
  ifCall=((p->Compute)==&Call);

  rebate=((ptOpt->Rebate.Val.V_NUMFUNC_1)->Compute)((ptOpt-
    >Rebate.Val.V_NUMFUNC_1)->Par,ptMod->T.Val.V_DATE);


  if ((ptOpt->DownOrUp).Val.V_BOOL==DOWN)
    upordown=0;
  else upordown=1;

  res = backwardfourier_vg_downout(ptOpt->EuOrAm.Val.V_BOOL
    ,upordown, ifCall, spot,ptMod->Sigma.Val.V_PDOUBLE,ptMod->
    Theta.Val.V_PDOUBLE,ptMod->Kappa.Val.V_PDOUBLE,
                     r, divid,
                     ptOpt->Maturity.Val.V_DATE-ptMod->T.
    Val.V_DATE, Met->Par[1].Val.V_DOUBLE, strike,
                     limit,rebate,
                     Met->Par[0].Val.V_DOUBLE, Met->Par[2]
    .Val.V_INT2,
```

```
                        &(Met->Res[0].Val.V_DOUBLE), &(Met->
    Res[1].Val.V_DOUBLE));

  return res;

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

  if ((opt->OutOrIn).Val.V_BOOL==OUT)
    if ((opt->Parisian).Val.V_BOOL==WRONG)
      if ((opt->EuOrAm).Val.V_BOOL==EURO)
        return  OK;

  return WRONG;
}


#endif //PremiaCurrentVersion
static int MET(Init)(PricingMethod *Met,Option *Opt)
{
  static int first=1;

  if (first)
    {
      Met->Par[0].Val.V_PDOUBLE=2.0;
      Met->Par[1].Val.V_PDOUBLE=0.001;
      Met->Par[2].Val.V_INT2=100;

      first=0;
    }
  return OK;
}

PricingMethod MET(AP_backwardfourierbar_vg)=
{
  "AP_BackwardFourierBar_VG",
  { {"Scale of logprice range", DOUBLE, {100}, ALLOW},
```

```
      {"Space Discretization Step",DOUBLE,{500},ALLOW},
      {"TimeStepNumber",INT2,{100},ALLOW},
      {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CALC(AP_backwardfourierbar_vg),
  {{"Price",DOUBLE,{100},FORBID},
      {"Delta",DOUBLE,{100},FORBID},
      {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CHK_OPT(AP_backwardfourierbar_vg),
  CHK_split,
  MET(Init)
};
```

References