Help

```
#include <stdlib.h>
#include  "variancegamma1d_std.h"
#include "math/wienerhopf.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2011+2) //The "#els
static int CHK_OPT(AP_backwardfourierdig_vg)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(AP_backwardfourierdig_vg)(void *Opt, void *Mod, PricingMethod *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else

/*////////////////////////////////////////////////////*/
static int ap_backwardfourier_amerdigital(double Spot, double sigma, double thet
    double r, double divid,
    double T, double h, double Strike1,
    double rebate,
    double er, long int step,
    double *ptprice, double *ptdelta)
{
  double  ptprice1, ptdelta1, mu, qu, om;
  double lm1, lp1, num = 1., nup = 1., cm, cp;

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

  int upordown = 1;
  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 *

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

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

  bi_barr(mu, qu, om, upordown, 2, Spot, lm1, lp1,
          num, nup, cm, cp, r, divid,
          T, h, Strike1, Strike1, rebate,
          er, step, &ptprice1, &ptdelta1);

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

  return OK;
}


//=============================================================================
int CALC(AP_backwardfourierdig_vg)(void *Opt, void *Mod, PricingMethod *Met)
{
  TYPEOPT *ptOpt = (TYPEOPT *)Opt;
  TYPEMOD *ptMod = (TYPEMOD *)Mod;
```

```
  double r, divid, strike, spot, rebate;

  NumFunc_1 *p;
  int res;

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

  rebate = p->Par[1].Val.V_DOUBLE;


  res = ap_backwardfourier_amerdigital(spot, ptMod->Sigma.Val.V_PDOUBLE, ptMod->
                                       r, divid,
                                       ptOpt->Maturity.Val.V_DATE - ptMod->T.Val
                                       Met->Par[0].Val.V_DOUBLE, Met->Par[2].Val
                                       &(Met->Res[0].Val.V_DOUBLE), &(Met->Res[1

  return res;

}

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

  if ((strcmp(((Option *)Opt)->Name, "DigitAmer") == 0))
    return  OK;

  return WRONG;
}


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

```
  if (first)
    {
      Met->HelpFilenameHint = "AP_backwardfourierdig_vg";
      Met->Par[0].Val.V_PDOUBLE = 2.0;
      Met->Par[1].Val.V_PDOUBLE = 0.01;
      Met->Par[2].Val.V_INT2 = 600;

      first = 0;
    }

  return OK;
}

PricingMethod MET(AP_backwardfourierdig_vg) =
{
  "AP_BackwardFourierDig_VG",
  { {"Scale of logprice range", DOUBLE, {100}, ALLOW},
    {"Space Discretization Step", DOUBLE, {500}, ALLOW},
    {"TimeStepNumber", INT2, {100}, ALLOW},
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CALC(AP_backwardfourierdig_vg),
  { {"Price", DOUBLE, {100}, FORBID},
    {"Delta", DOUBLE, {100}, FORBID},
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CHK_OPT(AP_backwardfourierdig_vg),
  CHK_split,
  MET(Init)
};
```