```
   Help
#include <stdlib.h>
#include  "nig1d_std.h"
#include "math/wienerhopf.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2009+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_fastwhamerdig_nig)(void *Opt, void *
    Mod)
{
  return NONACTIVE;
}
int CALC(AP_fastwhamerdig_nig)(void*Opt,void *Mod,Pricing
    Method *Met)
{
return AVAILABLE_IN_FULL_PREMIA;
}
#else

/*////////////////////////////////////////////////////*/
 static int wh_nig_amerdigital(double Spot, double sigma,
    double theta,double kappa,
            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, nup, cm, cp;

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

int upordown=1;

  alfa=sqrt(theta*theta+sig2/kappa)/sig2;
  beta=theta/sig2;
  cp=sigma/sqrt(kappa);
  cm=cp;
```

```
  lp1=alfa+beta;
  lm1=beta - alfa;
  nup=1.0;
  num=1.0;

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

  mu=r-divid+cp*(pow(alfa*alfa-(beta+1)*(beta+1), 0.5) -
    pow(alfa*alfa-beta*beta, 0.5));

  qu = r + cp*(pow(alfa*alfa-(beta+om)*(beta+om), 0.5) -
    pow(alfa*alfa-beta*beta, 0.5));

  fastwienerhopf(2, mu, qu, om, 0, 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_fastwhamerdig_nig)(void *Opt,void *Mod,Pricing
    Method *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 = wh_nig_amerdigital(spot,ptMod->Sigma.Val.V_PDOUBLE,
    ptMod->Theta.Val.V_DOUBLE,ptMod->Kappa.Val.V_SPDOUBLE,
        r, divid,
        ptOpt->Maturity.Val.V_DATE-ptMod->T.Val.V_DATE,
    Met->Par[1].Val.V_DOUBLE, strike,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_fastwhamerdig_nig)(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->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_fastwhamerdig_nig)=
{
  "AP_FastWHDig_Nig",
  { {"Scale of logprice range", DOUBLE, {100}, ALLOW},
    {"Space Discretization Step",DOUBLE,{500},ALLOW},
    {"TimeStepNumber",INT2,{100},ALLOW},
   {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CALC(AP_fastwhamerdig_nig),
  {{"Price",DOUBLE,{100},FORBID},
   {"Delta",DOUBLE,{100},FORBID},
   {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CHK_OPT(AP_fastwhamerdig_nig),
  CHK_split,
  MET(Init)
};
```

References