Help

```
#include <stdlib.h>
#include <math.h>
#include "pnl/pnl_vector.h"
#include "pnl/pnl_fft.h"
#include "math/wienerhopf.h"
#include "kou1d_pad.h"

#include "pnl/pnl_cdf.h"
#include"pnl/pnl_random.h"
#include"pnl/pnl_specfun.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2012+2) //The "#els
static int CHK_OPT(AP_WH_KOU_FloatingLookback)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(AP_WH_KOU_FloatingLookback)(void *Opt, void *Mod, PricingMethod *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else
/*////////////////////////////////////////////////////*/

//=============================================================================


static int ap_wienerhopf_lookbackfloating_kou(double s_maxmin, NumFunc_2 *P, dou
    double r, double divid, double sigma, double lambda, double lambdap, double

{
  int ifCall;
  double cp, cm, ptprice1, ptdelta1, mu, qu, omega, sig2, lp, lm;

  lp = lambdam;
  lm = -lambdap;

  /* if(ifCall==1)       //CALL//
    {omega=lm<-2. ? 2. : (-lm+1.)/2.;  }
    else                //PUT//
```

```
    {omega= lp>1. ? -1. : -lp/2.; }*/

  omega = 0;
  cp = (1 - P0) * lambda;
  cm = P0 * lambda;

  sig2 = sigma * sigma;

  mu = r - divid + cp / (lp + 1.0) + cm / (lm + 1.0) - sig2 / 2.0;

  qu = r - mu * omega - sig2 * omega * omega / 2 + cp + cm - cp * lp / (lp + ome

//CALL
  if ((P->Compute) == &Call_StrikeSpot2)
    {
      ifCall = 1;
    }
//PUT
  if ((P->Compute) == &Put_StrikeSpot2)
    {
      ifCall = 0;
    }

  lookback_fls(4, mu, qu, omega, ifCall, Spot, s_maxmin, lm, lp,
               sigma, sigma, cm, cp, r, divid,
               T, h, er, &ptprice1, &ptdelta1);

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

  return OK;
}

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

```
  r = log(1. + ptMod->R.Val.V_DOUBLE / 100.);
  divid = log(1. + ptMod->Divid.Val.V_DOUBLE / 100.);

  return  ap_wienerhopf_lookbackfloating_kou((ptOpt->PathDep.Val.V_NUMFUNC_2)->P
          ptOpt->PayOff.Val.V_NUMFUNC_2, ptMod->S0.Val.V_PDOUBLE, ptOpt->Maturit
          r, divid, ptMod->Sigma.Val.V_PDOUBLE, ptMod->Lambda.Val.V_PDOUBLE, ptM
          ptMod->LambdaMinus.Val.V_PDOUBLE, ptMod->P.Val.V_PDOUBLE,
          Met->Par[1].Val.V_SPDOUBLE, Met->Par[0].Val.V_SPDOUBLE, &(Met->Res[0].
}


static int CHK_OPT(AP_WH_KOU_FloatingLookback)(void *Opt, void *Mod)
{
  if ((strcmp(((Option *)Opt)->Name, "LookBackCallFloatingEuro") == 0) || (strcm
    return OK;
  return WRONG;
}


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

  if (first)
    {
      Met->HelpFilenameHint = "AP_FASTWH_KOU_LookbackFloating";
      Met->Par[0].Val.V_PDOUBLE = 2.0;
      Met->Par[1].Val.V_PDOUBLE = 0.001;

      first = 0;
    }
  return OK;
}

PricingMethod MET(AP_WH_KOU_FloatingLookback) =
{
  "AP_FastWH",
  { {"Scale of logprice range", DOUBLE, {100}, ALLOW},
    {"Space Discretization Step", DOUBLE, {500}, ALLOW},
```

```
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CALC(AP_WH_KOU_FloatingLookback),
  { {"Price", DOUBLE, {100}, FORBID},
    {"Delta", DOUBLE, {100}, FORBID},
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CHK_OPT(AP_WH_KOU_FloatingLookback),
  CHK_split,
  MET(Init)
};
```