

Help

```

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

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

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

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

static int ap_wienerhopf_lookbackfixed_kou(double s_maxmin,
    NumFunc_2*P, double Spot, double Strike, double T,
    double r,double divid,double sigma,double lambda_d,
    double lambda_p,double lambda_m,double P0, double h,
    double er, double *ptprice, double *ptdelta)
{
    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+omega)-
        cm*lm/(lm+omega);

//CALL
    if ((P->Compute)==&Call_OverSpot2)
    {
        ifCall=1;
    }
//PUT
    if ((P->Compute)==&Put_OverSpot2)
    {
        ifCall=0;
    }

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

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

```

```

    return OK;
}

//=====
=====
int CALC(AP_WH_KOU_FixedLookback)(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_lookbackfixed_kou((ptOpt->PathDep.
        Val.V_NUMFUNC_2)->Par[4].Val.V_PDOUBLE,
        ptOpt->PayOff.Val.V_NUMFUNC_2,ptMod->S0.Val.V_PDOUBLE,
        (ptOpt->PathDep.Val.V_NUMFUNC_2)->Par[3].Val.V_DOUBLE,pt
        Opt->Maturity.Val.V_DATE-ptMod->T.Val.V_DATE,
        r,divid,ptMod->Sigma.Val.V_PDOUBLE,ptMod->Lambda.Val.
        V_PDOUBLE,ptMod->LambdaPlus.Val.V_PDOUBLE,
        ptMod->LambdaMinus.Val.V_PDOUBLE,ptMod->P.Val.V_PDOUB
        LE,
        Met->Par[1].Val.V_SPDOUBLE,Met->Par[0].Val.V_SPDOUBLE,
        &(Met->Res[0].Val.V_DOUBLE), &(Met->Res[1].Val.V_DOUBLE));
}

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

#endif //PremiaCurrentVersion

```

```

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

    if (first)
    {
        Met->HelpFilenameHint = "AP_WH_KOU_FixedLookback";
        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_FixedLookback)=
{
    "AP_FastWH",
    { {"Scale of logprice range", DOUBLE, {100}, ALLOW},
      {"Space Discretization Step",DOUBLE,{500},ALLOW},
      {" ",PREMIA_NULLTYPE,{0},FORBID}},
    CALC(AP_WH_KOU_FixedLookback),
    {{"Price",DOUBLE,{100},FORBID},
      {"Delta",DOUBLE,{100},FORBID},
      {" ",PREMIA_NULLTYPE,{0},FORBID}},
    CHK_OPT(AP_WH_KOU_FixedLookback),
    CHK_split,
    MET(Init)
};

```

References