

Help

```

#include <stdlib.h>
#include <math.h>
#include "pnl/pnl_vector.h"
#include "pnl/pnl_fft.h"
#include "math/wienerhopf.h"
#include "cgmy1d_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 av
    ailable after the (year of creation of this file + 2)
static int CHK_OPT(AP_WH_FloatingLookback)(void *Opt, void
    *Mod)
{
    return NONACTIVE;
}
int CALC(AP_WH_FloatingLookback)(void*Opt,void *Mod,Pricing
    Method *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
#else
/*////////////////////////////////////*/

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

static int ap_wienerhopf_lookbackfloating(double s_maxmin,
    NumFunc_2*P, double Spot, double T,
    double r,double divid,double C,double G,double
    M,double Y,double h, double er, double *ptprice, double *pt
    delta)

{
    double cnu, lp1, lm1, lpnu, lmnu, ptprice1, ptdelta1, mu
    , qu, om;

```

```

int ifCall;

    if(M<2 || G<=1 || Y>=2 || Y==0)
    {
        printf("Invalid parameters. We must have M>=2, G>1, 0<
Y<2{n");
    }
//CALL
    if ((P->Compute)==&Call_StrikeSpot2)
    {
        ifCall=1;
    }
//PUT
    if ((P->Compute)==&Put_StrikeSpot2)
    {
        ifCall=0;
    }

    lm1=-M;
    lp1=G;

    om=0.0;

    cnu=C*tgamma(-Y);

    lpnu=exp(Y*log(lp1));
    lmnu=exp(Y*log(-lm1));

    mu= r - divid + cnu*(lpnu-exp(Y*log(lp1+1.0))) + cnu*(
    lmnu-exp(Y*log(-lm1-1.0)));

    qu = r + (pow(lp1,Y) - pow(lp1+om,Y))*cnu + (pow(-lm1,Y)
    -pow(-lm1-om,Y))*cnu;

    lookback_fls(1, mu, qu, om, ifCall, Spot, s_maxmin, lm1,

```

```

        lp1,
            Y, Y, cnu, cnu, r, divid,
            T, h, er, &ptprice1, &ptdelta1);

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

return OK;
}

//=====
=====
int CALC(AP_WH_FloatingLookback)(void*Opt,void *Mod,Pricing
    Method *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((ptOpt->PathDep.
        Val.V_NUMFUNC_2)->Par[4].Val.V_PDOUBLE,
        ptOpt->PayOff.Val.V_NUMFUNC_2,ptMod->S0.Val.V_PDOUBLE,
        ptOpt->Maturity.Val.V_DATE-ptMod->T.Val.V_DATE,
        r,divid,ptMod->C.Val.V_PDOUBLE,ptMod->G.Val.V_DOUBLE,
        ptMod->M.Val.V_SPDOUBLE,ptMod->Y.Val.V_PDOUBLE,
        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_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_WH_FloatingLookback";
        Met->Par[0].Val.V_PDOUBLE=1.0;
        Met->Par[1].Val.V_PDOUBLE=0.001;

        first=0;
    }
    return OK;
}

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

```

References