

Help

```

extern "C"{
#include "kou1d_lim.h"
}
#include<iostream>
#include<stdlib.h>
#include<cmath>
#include"math/ap_kou_model/functions.h"

extern "C"{

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
    (2008+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_Kou_In)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
int CALC(AP_Kou_In)(void*Opt,void *Mod,PricingMethod *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
#else

static int Kou_Ap_In(int b_type,double l,double rebate,
    double S0,NumFunc_1 *P,double T,double r,double divid,double sigma,
    double lambda,double lambdap,double lambdam,double p,
    double *ptPrice,double *ptDelta)
{
    int op_type=1;
    long double ksi, cst1, cst2, dcst1, dcst2, proba,dprob
    a,temp,dptPrice;
    long double x[9];
    double h=0.01;

    /*Call Case*/
    if ((P->Compute)==&Call)
        op_type=0;
    else
        /*Put Case*/
        if ((P->Compute)==&Put)

```

```

        op_type=1;

        double K=P->Par[0].Val.V_DOUBLE;

        ksi=p*lambdap/(lambdap-1)+(1-p)*lambdam/(lambdam+1)-1;
        if(b_type==0)//down
        {
            x[0]=-((r-divid)-sigma*sigma/2-lambda*ksi);
            x[1]=sigma;
            x[2]=lambda;
            x[3]=(1-p);
            x[4]=lambdam;
            x[5]=lambdap;
            x[6]=log(S0/K);
            x[7]=log(S0/l);
            x[8]=T;
            if(op_type==0)//call
            {
                proba=psiM(x,T);
                cst1=proba-psiB(x,T);
                x[6]=log((S0+h)/K);
                x[7]=log((S0+h)/l);
                dproba=psiM(x,T);
                dcst1=dproba-psiB(x,T);
            }
            else//put
            {
                proba=psiM(x,T);
                cst1=psiB(x,T);
                x[6]=log((S0+h)/K);
                x[7]=log((S0+h)/l);
                dproba=psiM(x,T);
                dcst1=psiB(x,T);
            }
            x[0]=-((r-divid)+sigma*sigma/2-lambda*ksi);
            x[2]=lambda*(ksi+1);
            x[3]=p*lambdap/((1+ksi)*(lambdap-1));x[3]=1-x[3];
            x[4]=lambdap-1;
            x[5]=lambdam+1;
            temp=x[4];
            x[4]=x[5];

```

```

x[5]=temp;
if(op_type==0)//call
{
    dcst2=psiM(x,T)-psiB(x,T);
    x[6]=log(S0/K);
    x[7]=log(S0/l);
    cst2=psiM(x,T)-psiB(x,T);

    *ptPrice=S0*exp(-divid*T)*cst2-K*exp(-r*T)*cst1+exp(-
r*T)*rebate*(1-proba);
    dptPrice=(S0+h)*exp(-divid*T)*dcst2-K*exp(-r*T)*dcst1+
exp(-r*T)*rebate*(1-dproba);
    *ptDelta=(dptPrice-*ptPrice)/h;
}
else//put
{
    dcst2=psiB(x,T);
    x[6]=log(S0/K);
    x[7]=log(S0/l);
    cst2=psiB(x,T);

    *ptPrice=K*exp(-r*T)*cst1-S0*exp(-divid*T)*cst2+exp(-
r*T)*rebate*(1-proba);
    dptPrice=K*exp(-r*T)*dcst1-(S0+h)*exp(-divid*T)*dcst2+
exp(-r*T)*rebate*(1-dproba);
    *ptDelta=(dptPrice-*ptPrice)/h;
}
}
else//up
{
    x[0]=(r-divid)-sigma*sigma/2-lambda*ksi;
    x[1]=sigma;
    x[2]=lambda;
    x[3]=p;
    x[4]=lambdap;
    x[5]=lambdam;
    x[6]=log(K/S0);
    x[7]=log(l/S0);
    x[8]=T;
    if(op_type==0)//call
    {

```

```

        proba=psiM(x,T);
        cst1=psiB(x,T);
        x[6]=log(K/(S0+h));
        x[7]=log(1/(S0+h));
        dproba=psiM(x,T);
        dcst1=psiB(x,T);
    }
    else//put
    {
        proba=psiM(x,T);
        cst1=proba-psiB(x,T);
        x[6]=log(K/(S0+h));
        x[7]=log(1/(S0+h));
        dproba=psiM(x,T);
        dcst1=dproba-psiB(x,T);
    }
    x[0]=(r-divid)+sigma*sigma/2.-lambda*ksi;
    x[2]=lambda*(ksi+1.);
    x[3]=p*lambda*p/((1.+ksi)*(lambda*p-1.));
    x[4]=lambda*p-1.;
    x[5]=lambda*m+1.;
    if(op_type==0)//call
    {
        dcst2=psiB(x,T);
        x[6]=log(K/S0);
        x[7]=log(1/S0);
        cst2=psiB(x,T);

        *ptPrice=S0*exp(-divid*T)*cst2-K*exp(-r*T)*cst1+exp(-
r*T)*rebate*(1-proba);
        dptPrice=(S0+h)*exp(-divid*T)*dcst2-K*exp(-r*T)*dcst1+
exp(-r*T)*rebate*(1-dproba);
        *ptDelta=(dptPrice-*ptPrice)/h;
    }
    else//put
    {
        dcst2=psiM(x,T)-psiB(x,T);
        x[6]=log(K/S0);
        x[7]=log(1/S0);
        cst2=psiM(x,T)-psiB(x,T);
    }

```

```

        *ptPrice=K*exp(-r*T)*cst1-S0*exp(-divid*T)*cst2+exp(-
r*T)*rebate*(1-proba);
        dptPrice=K*exp(-r*T)*dcst1-(S0+h)*exp(-divid*T)*dcst2+
exp(-r*T)*rebate*(1-dproba);
        *ptDelta=(dptPrice-*ptPrice)/h;
    }
}

return OK;
}

int CALC(AP_Kou_In)(void*Opt,void *Mod,PricingMethod *
Met)
{
    TYPEOPT* ptOpt=(TYPEOPT*)Opt;
    TYPEMOD* ptMod=(TYPEMOD*)Mod;
    double r,divid,limit,rebate;
    int upordown;

    r=log(1.+ptMod->R.Val.V_DOUBLE/100.);
    divid=log(1.+ptMod->Divid.Val.V_DOUBLE/100.);
    limit=((ptOpt->Limit.Val.V_NUMFUNC_1)->Compute)((pt
Opt->Limit.Val.V_NUMFUNC_1)->Par,ptMod->T.Val.V_DATE);
    rebate=((ptOpt->Rebate.Val.V_NUMFUNC_1)->Compute)((pt
Opt->Rebate.Val.V_NUMFUNC_1)->Par,ptMod->T.Val.V_DATE);

    if ((ptOpt->DownOrUp).Val.V_BOOL==DOWN)
        upordown=0;
    else upordown=1;

    return Kou_Ap_In(upordown,limit,rebate,ptMod->S0.Val.
V_PDOUBLE,ptOpt->PayOff.Val.V_NUMFUNC_1,ptOpt->Maturity.Val
.V_DATE-ptMod->T.Val.V_DATE,r,divid,ptMod->Sigma.Val.V_PDO
UBLE,ptMod->Lambda.Val.V_PDOUBLE,ptMod->LambdaPlus.Val.V_
PDOUBLE,ptMod->LambdaMinus.Val.V_PDOUBLE,ptMod->P.Val.V_PDO
UBLE,&(Met->Res[0].Val.V_DOUBLE),&(Met->Res[1].Val.V_
DOUBLE));
}

static int CHK_OPT(AP_Kou_In)(void *Opt, void *Mod)
{

```

```

Option* ptOpt=(Option*)Opt;
TYPEOPT* opt=(TYPEOPT*)(ptOpt->TypeOpt);

/*if ((opt->RebOrNo).Val.V_BOOL==NOREBATE)*/
    if ((opt->OutOrIn).Val.V_BOOL==IN)
        if ((opt->EuOrAm).Val.V_BOOL==EURO)
            if ((opt->Parisian).Val.V_BOOL==WRONG)
                return OK;

    return WRONG;
}

#endif //PremiaCurrentVersion
static int MET(Init)(PricingMethod *Met,Option *Mod)
{
    return OK;
}

PricingMethod MET(AP_Kou_In)=
{
    "AP_Kou_Barrier_In",
    {" " ,PREMIA_NULLTYPE,{0},FORBID}},
    CALC(AP_Kou_In),
    {"Price",DOUBLE,{100},FORBID},{ "Delta",DOUBLE,{100},FORBID},{ " " ,PREMIA_NULLTYPE,{0},FORBID}},
    CHK_OPT(AP_Kou_In),
    CHK_ok,
    MET(Init)
} ;
}

```

References