

## [Help](#)

```
extern "C" {
#include "
href../../mod/kou1d/kou1d_pad/kou1d_pad_h_src.pdfkou1d_pad.h"
#include "
href../../common/error_msg_h_src.pdferror_msg.h"
}
#include"
href../../common/math/ap_kou_model/functions_h_src.pdfmath/ap_kou_model/funct

extern "C" {
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2008+2) //The "#els
    static int CHK_OPT(AP_Kou_Fixed)(void *Opt, void *Mod)
    {
        return NONACTIVE;
    }
    int CALC(AP_Kou_Fixed)(void *Opt, void *Mod, PricingMethod *Met)
    {
        return AVAILABLE_IN_FULL_PREMIA;
    }
#else
    static int Kou_Ap_Fixed(double s_maxmin, NumFunc_2 *P, double S0, double T, do
    {
        double K;
        long double x[11];
        double ksi = p * lambdap / (lambdap - 1) + (1 - p) * lambdam / (lambdam + 1)
        K = P->Par[0].Val.V_DOUBLE;

        /*Call Case*/
        if ((P->Compute) == &Call_OverSpot2)
        {
            if ((s_maxmin >= S0) || (K >= S0))
            {
                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] = S0;
```

```

        x[7] = r;
        x[8] = T;
        x[9] = (s_maxmin > K) ? s_maxmin : K;
        x[10] = divid;

        *ptprice = PLB(x, T) + S0 * exp(-divid * T) - K * exp(-r * T);
        *ptdelta = dPLB(x, T) + exp(-divid * T);
    }
    else return UNTREATED_CASE;

}

else if ((P->Compute) == &Put_OverSpot2)
{
    if ((s_maxmin <= S0) || (K <= S0))
    {
        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] = S0;
        x[7] = r;
        x[8] = T;
        x[9] = (s_maxmin < K) ? s_maxmin : K;
        x[10] = divid;

        *ptprice = CLB(x, T) - S0 * exp(-divid * T) + K * exp(-r * T);
        *ptdelta = dCLB(x, T) - exp(-divid * T);
    }
    else return UNTREATED_CASE;
}

return OK;
}

int CALC(AP_Kou_Fixed)(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 Kou_Ap_Fixed((ptOpt->PathDep.Val.V_NUMFUNC_2)->Par[4].Val.V_PDOUBLE,
}

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

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

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

```