

## [Help](#)

```
extern "C" {
#include "
href../../../../mod/rstemperedstable1d/rstemperedstable1d_std/rstemperedstable1d_st
}

#include "
href../../../../common/math/numerics_h_src.pdfmath/numerics.h"

extern "C" {

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2008+2) //The "#els
    static int CHK_OPT(AP_IAC)(void *Opt, void *Mod)
    {
        return NONACTIVE;
    }
    int CALC(AP_IAC)(void *Opt, void *Mod, PricingMethod *Met)
    {
        return AVAILABLE_IN_FULL_PREMIA;
    }
#else

    static const double eps = 0.00000001;
    static const double mpi = 3.14159265358;
    double dt, mu, teta;
    static double inteput(double x, double lm, double lp,
                          double alm, double alp,
                          double cnum, double cnup,
                          double r, double eps0);
    static double run_nu2(double u, double v1, double v2,
                          double lam1, double lam2, double nu1, double nu2,
                          double rho, double eps0);
    static double nufun(double x, double uvpar[10]);

    /*////////////////////////////////////////*/
    int iac_kobol_europut(int ifCall, double lm, double lp,
                        double alm, double alp, double cm, double cp,
                        double r, double T, double Strike,
                        double Spot, double eps, double *Price)
```

```

{

    //////////////////////////////////////
    double lpnu, lmnu;
    double cnu, cnum;
    /* swap=lp;
       lp=lm;
       lm=swap;
    */
    dt = T; ///Nt;

    lpnu = pow(lp, alp);
    lmnu = pow(lm, alm);
    cnu = cp * pnl_tgamma(-alp);
    cnum = cm * pnl_tgamma(-alm);

    mu = r + cnu * (lpnu - pow(lp + 1, alp)) + cnum * (lmnu - pow(lm - 1, alm))
    //-----

    *Price = Strike * inteput(log(Spot / Strike), lm, lp, alm, alp, cnum, cnu,
    if (ifCall)
    {
        *Price = *Price + Spot - Strike * exp(-r * T);
        // *Delta = *Delta + exp(-divid*T);
    }
    //////////////////////////////////////

    return OK;
}

////////////////////////////////////integration
static double inteput(double x, double lm, double lp,
                     double alm, double alp,
                     double cnum, double cnu,
                     double r, double eps0)
{
    double factr, res, intres, term = 0;
    double etaur, fac;

```

```

double lpnu, lmnu;
double uq;

lpnu = pow(lp, alp);
lmnu = pow(lm, alm);

etaur = exp(-dt * r);

fac = exp(-dt * (r + cnup * lpnu + cnum * lmnu)) / mpi;

uq = x + dt * mu;

if (uq >= 0)
{
    factr = fac * exp(-uq * lp) / alp;
    intres = run_nu2(uq, -dt * cnup, -dt * cnum, lp, lm, alp, alm, 1.0, eps0);
    term = 0;
}
else
{
    factr = fac * exp(uq * lm) / alm;
    intres = run_nu2(-uq, -dt * cnum, -dt * cnup, lm, lp, alm, alp, -1.0, eps0);
    term = etaur - exp(x);
};
res = term + factr * intres;
return res;
}

/*////////////////////////////////////////*/
static double run_nu2(double u, double v1, double v2,
                    double lam1, double lam2, double nu1, double nu2,
                    double rho, double eps0)
{
    double h, s1, s2, intv1, intv2, x, intg1, intg2;
    double lambda, anu, bnu;
    long points1, n, i;
    double uvpar[10];

```

```

anu = cos(mpi * nu1);
bnu = sin(mpi * nu1);
uvpar[0] = u;
uvpar[1] = v1;
uvpar[2] = v2;
uvpar[3] = lam1;
uvpar[4] = lam2;
uvpar[5] = nu1;
uvpar[6] = nu2;
uvpar[7] = rho;
uvpar[8] = anu;
uvpar[9] = bnu;

lambda = 1;
intg1 = 1;
intg2 = 0;
points1 = 0;
while ((intg2 == 0) || (fabs(intg1 - intg2) > eps0 * intg2))
{
    h = lambda / 2.0;
    s1 = nufun(lambda, uvpar);      /*sin(lambda*vnu);
    s2 = nufun(h, uvpar);          /*sin(h*vnu);
    intv2 = h * (s1 + 4.0 * s2) / 3.0;
    intv1 = 0;

    n = 2;

    while ((intv2 == 0) || (fabs(intv1 - intv2) > eps0 * intv2))
    {
        intv1 = intv2;
        s1 += 2.0 * s2;
        s2 = 0;
        x = h / 2.0;
        for (i = 1; i <= n; i++)
        {
            s2 += nufun(x, uvpar); /*sin(vnu*x);
            x += h;
        }
        h /= 2.0;
    }
}

```

```

        n *= 2;
        intv2 = h * (s1 + 4.0 * s2) / 3.0;
    };

    n = (long int)ceil(n / 2.0) + 1;
    points1 += n;
    intg1 = intg2;
    intg2 = intv2;
// lamm=lambda;
    lambda += 1;
}
return intg1;
}
////////////////////////////////////nufun
static double nufun(double x, double uvpar[10])
{
    double teta, z, z1, res;
    double u = uvpar[0];
    double v1 = uvpar[1];
    double v2 = uvpar[2];
    double lam1 = uvpar[3];
    double lam2 = uvpar[4];
    double nu1 = uvpar[5];
    double nu2 = uvpar[6];
    double rho = uvpar[7];
    double aa = uvpar[8];
    double bb = uvpar[9];

    teta = 1 / nu1;
    z = exp(teta * log(x));
    z1 = exp((teta - 1) * log(x));

    res = exp(-u * z) * z1 * sin(v1 * x * bb) / (z + lam1) / (z + lam1 + rho) *
    return res;
}

/*////////////////////////////////////*/

int CALC(AP_IAC)(void *Opt, void *Mod, PricingMethod *Met)
{
    TYPEOPT *ptOpt = (TYPEOPT *)Opt;

```

```

    TYPEMOD *ptMod = (TYPEMOD *)Mod;
    double r, strike, spot;
    NumFunc_1 *p;

    r = log(1. + ptMod->R.Val.V_DOUBLE / 100.);
    //divid=log(1.+ptMod->Divid.Val.V_DOUBLE/100.);
    p = ptOpt->PayOff.Val.V_NUMFUNC_1;
    strike = p->Par[0].Val.V_DOUBLE;
    spot = ptMod->S0.Val.V_DOUBLE;

    return iac_kobol_europut((p->Compute) == &Call,
                            ptMod->LambdaPlus.Val.V_DOUBLE, ptMod->LambdaMinus.
                            ptMod->AlphaPlus.Val.V_RGDOUBLE, ptMod->AlphaMinus.
                            ptMod->CPlus.Val.V_DOUBLE, ptMod->CMinus.Val.V_DOUB
                            r,
                            ptOpt->Maturity.Val.V_DATE - ptMod->T.Val.V_DATE,
                            strike, spot, eps,
                            &(Met->Res[0].Val.V_DOUBLE)/*PRICE*/);

}

static int CHK_OPT(AP_IAC)(void *Opt, void *Mod)
{
    if ((strcmp(((Option *)Opt)->Name, "PutEuro") == 0) || (strcmp(((Option *)Op
        return OK;

    return WRONG;
}

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

PricingMethod MET(AP_IAC) =
{
    "AP_IAC",
    {" " , PREMIA_NULLTYPE, {0}, FORBID}},
    CALC(AP_IAC),

```

```

    { {"Price", DOUBLE, {100}, FORBID},
      {" ", PREMIA_NULLTYPE, {0}, FORBID}
    },
    CHK_OPT(AP_IAC),
    CHK_ok ,
    MET(Init)
} ;

/*////////////////////////////////////////*/
}

```