

[Help](#)

```
extern"C" {
#include "hes1d_vol.h"
#include "numfunc.h"
}

#include "math/intg.h"

extern "C" {

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2008+2) //The "#els
    static int CHK_OPT(AP_HES_VOLATILITYSWAP2)(void *Opt, void *Mod)
    {
        return NONACTIVE;
    }
    int CALC(AP_HES_VOLATILITYSWAP2)(void *Opt, void *Mod, PricingMethod *Met)
    {
        return AVAILABLE_IN_FULL_PREMIA;
    }
#else
    /*////////////////////////////////////*/
    static int
    ap_hes_volswap2(double sigma0, double ka, double theta, double sigma2,
                    double rhow, double r, double divid, double T, double Strike,
                    double Spot, double *fairval, double *Price)
    {

        double eVar, eVol, varVol2, ekt, ekt2, sig;

        ka *= T;
        ekt = exp(-ka);
        ekt2 = ekt * ekt;
        eVar = theta + (sigma0 - theta) * (1.0 - ekt) / ka;

        sig = sigma2 * sigma2;
        eVar = sqrt(eVar);
        varVol2 = T * sig / ka / ka / ka * ((1.0 - 2.0 * ekt * ka - ekt2) * (sigma0
        eVol = eVar - varVol2 / eVar / eVar / eVar / 8.0;
        //fair strike of volatility swap
        *fairval = eVol * 100;
```

```

    // price of vol swap
    *Price = exp(-r * T) * (*fairval - Strike);
    return OK;
}

/*-----*/

int CALC(AP_HES_VOLATILITYSWAP2)(void *Opt, void *Mod, PricingMethod *Met)
{
    TYPEOPT *ptOpt = (TYPEOPT *)Opt;
    TYPEMOD *ptMod = (TYPEMOD *)Mod;
    double r, divid, 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 ap_hes_volswap2(
        ptMod->Sigma0.Val.V_PDOUBLE,
        ptMod->MeanReversion.Val.V_PDOUBLE,
        ptMod->LongRunVariance.Val.V_PDOUBLE,
        ptMod->Sigma.Val.V_PDOUBLE,
        ptMod->Rho.Val.V_PDOUBLE,
        r, divid,
        ptOpt->Maturity.Val.V_DATE - ptMod->T.Val.V_DATE,
        strike, spot,
        &(Met->Res[0].Val.V_DOUBLE)/*FAIRVAL*/,
        &(Met->Res[1].Val.V_DOUBLE)/*PRICE*/);
}

static int CHK_OPT(AP_HES_VOLATILITYSWAP2)(void *Opt, void *Mod)
{
    if ((strcmp(((Option *)Opt)->Name, "VolatilitySwap") == 0))
        return OK;

    return WRONG;
}

```

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

PricingMethod MET(AP_HES_VOLATILITYSWAP2) =
{
    "AP_HES_VOLATILITYSWAP2", //"2-terms approximation",
    { {" ", PREMIA_NULLTYPE, {0}, FORBID}},
    CALC(AP_HES_VOLATILITYSWAP2),
    { {"Fair strike in annual volatility points", DOUBLE, {100}, FORBID},
      {"Price ", DOUBLE, {100}, FORBID},
      {" ", PREMIA_NULLTYPE, {0}, FORBID}
    },
    CHK_OPT(AP_HES_VOLATILITYSWAP2),
    CHK_ok ,
    MET(Init)
} ;

}
```