

```

    Help
extern "C"{
#include "temperedstable1d_vol.h"
}
#include "math/numerics.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_CGMY_VOLATILITYSWAP)(void *Opt, void
    *Mod)
{
    return NONACTIVE;
}
int CALC(AP_CGMY_VOLATILITYSWAP)(void *Opt,void *Mod,PricingMethod *Met)
{
return AVAILABLE_IN_FULL_PREMIA;
}
#else

    static double replFun(double v, double m);

    int ap_cgmy_realvar(int ifCall, double S0, double Strike,
        double T, double r, double divid, double alphap, double
        alpham,double lambdap,double lambdam,double cp,double cm,
        double parsigma, double parstep, int exp2, double *ptprice);

    /*////////////////////////////////////*/
    static int ap_cgmy_volatilityswaps(double S0, double Strike,
        double T, double r, double divid, double ap, double am,
        double lap,double lam,double cpp,double cmm, double *ptfairval,
        double *ptprice)
    {

        double *replStrikes;
        double *replOptions;
        double *replWeights;
        int *CallPuts;

```

```
int flag;
double strikeshstep=100.0;
double pvfactor=exp(-r*T);

double gamma2p=tgamma(2.0-ap);
double gamma2m=tgamma(2.0-am);
double lpnu=exp((2.0-ap)*log(lap));
double lmnu=exp((2.0-am)*log(lam));

int k, k0, res, replN=10;
double optprice, tweight, tstrike, tprice;

//Expected realized variance
double mval=T*(cpp*gamma2p/lpnu+cmm*gamma2m/lmnu)*10000;

replStrikes = new double[replN];
replOptions = new double[replN];
replWeights = new double[replN];
CallPuts = new int[replN];

tprice=0;
tsstrike=mval;
k=0;
flag=1;
while((k<replN)&&(flag))
{
    replStrikes[k]=(k+1)*strikeshstep;
    CallPuts[k]=(mval<replStrikes[k]);
    flag=!CallPuts[k];
    k++;
}
k0=k-2;
for(;k<replN;k++)
{
    replStrikes[k]=(k+1)*strikeshstep;
    CallPuts[k]=1;
}

//weights for puts
tweight=0;
tsstrike=mval;
```

```

for(k=k0;k>=0;k--)
{
    replWeights[k]=( replFun(replStrikes[k], mval)-replF
un(tstrike, mval) )/strikestep - tweight;
    tweight+= replWeights[k];
    res=ap_cgmy_realvar(CallPuts[k], S0, sqrt(tstrike/T),
        T, r, divid, ap, am,lap, lam, cpp, cmm, 10.0, 0.25, 12, &
    optprice);
    if(res) {return 1;}
    replOptions[k]=optprice*optprice*T;
    tstrike = replStrikes[k];
    tprice += replOptions[k]*replWeights[k];
}
//weights for calls
tweight=0;
tsstrike=mval;
for(k=k0+1;k<replN;k++)
{
    replWeights[k]=( replFun(replStrikes[k], mval) - rep
lFun(tstrike, mval) )/strikestep - tweight;
    tweight+= replWeights[k];
    res=ap_cgmy_realvar(CallPuts[k], S0, sqrt(tstrike/T),
        T, r, divid, ap, am,lap, lam, cpp, cmm, 10.0, 0.25, 12, &
    optprice);
    if(res) {return 1;}
    replOptions[k]=optprice*optprice*T;
    tstrike = replStrikes[k];
    tprice+= replOptions[k]*replWeights[k];
}

tprice = sqrt(mval)-tprice;

//Fair strike
*ptfairval= tprice;
// price
*ptprice=pvfactor*(tprice-Strike);

delete [] replStrikes;
delete [] replOptions;
delete [] replWeights;

```

```

delete [] CallPuts;

return OK;
}

//-----
-----
static double replFun(double v, double m)
{
return (m+v)/2.0/sqrt(m)-sqrt(v);
}

//-----
-----

int CALC(AP_CGMY_VOLATILITYSWAP)(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_cgmy_volatilityswaps(
spot, strike, ptOpt->Maturity.Val.V_DATE-ptMod->T.Val
.V_DATE, r, divid, ptMod->AlphaPlus.Val.V_PDOUBLE, ptMod-
>AlphaMinus.Val.V_PDOUBLE, ptMod->LambdaPlus.Val.V_PDOUB
LE, ptMod->LambdaMinus.Val.V_PDOUBLE, ptMod->CPlus.Val.V_PDO
UBLE, ptMod->CMinus.Val.V_PDOUBLE,
&(Met->Res[0].Val.V_DOUBLE), &(Met->Res[1].Val.V_
DOUBLE));
}

static int CHK_OPT(AP_CGMY_VOLATILITYSWAP)(void *Opt, voi

```

```

    d *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_CGMY_VOLATILITYSWAP)=
{
    "AP_CGMY_VOLATILITYSWAP",
    {{" ",PREMIA_NULLTYPE,{0},FORBID}},
    CALC(AP_CGMY_VOLATILITYSWAP),
    { {"Fair strike value in annual volatility points",
    DOUBLE,{100},FORBID},
    {"Price ",DOUBLE,{100},FORBID},
    {" ",PREMIA_NULLTYPE,{0},FORBID}},
    CHK_OPT(AP_CGMY_VOLATILITYSWAP),
    CHK_ok ,
    MET(Init)
} ;

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

```

References