

## [Help](#)

```
#include "
href../../../../mod/hes1d/hes1d_stda/hes1d_stda_h_src.pdfhes1d_stda.h"
#include "
href../../../../common/enums_h_src.pdfenums.h"
#include "
href../../../../common/error_msg_h_src.pdferror_msg.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2015+2) //The "#els
static int CHK_OPT(AP_FOURIERCOSINE_GMIB_HES)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
int CALC(AP_FOURIERCOSINE_GMIB_HES)(void *Opt, void *Mod, PricingMethod *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
#else

static void Valomega(double a_global, double b_global, int N, /*input & output*/
{
    int i;
    for (i=0;i<N;i++)
pnl_vect_set(omega,i,((double)i)*M_PI/(b_global-a_global));
}

static void Valcf(double thetav, double T,double a_global,double b_global,double
{
    dcomplex D, G, temp1, temp2, temp3, temp4;
    double omegaj;
    double a = a_global - fee*T;
    int j;

    for (j=0;j<N;j++)
    {
        omegaj=pnl_vect_get(omega,j);

        D=Cpow_real(Cadd(Cpow_real(RCsub(kappav,Complex(0,rho*sigmav*omegaj)),2),CRmul
```

```

G=Cdiv(Csub(Complex(kappav,-rho*sigmav*omegaj),D),Cadd(Complex(kappav,-rho*sig

temp1=RCsub(1,Cexp(CRmul(D,-T)));
temp2=RCsub(1,Cmul(G,Cexp(CRmul(D,-T))));
temp3=Csub(Complex(kappav,-rho*sigmav*omegaj),D);
temp4=RCsub(1,G);

pnl_vect_complex_set(cf,j,Cmul(Cmul(Cexp(Cadd(Complex(0,omegaj*(r-fee)*T),CRmu

    }
}

static void cf0(/*input & output*/PnlVectComplex *cf)
{
    pnl_vect_complex_set_real(cf,0,0.5*pnl_vect_complex_get_real(cf,0));
    pnl_vect_complex_set_imag(cf,0,0.5*pnl_vect_complex_get_imag(cf,0));
}

static void Valvt(double a_global,double b_global,double alphav, double P, doubl
{
    int j;
    double omegaj;
    double a = a_global - fee*T;
    double b = b_global - fee*T;

    for (j=0;j<N;j++)
    {

        omegaj=pnl_vect_get(omega,j);
        if (j==0)
        {

            pnl_vect_set(V,0,(exp(a)-1.0-a)*(2.0/(b-a))*alphav*P*M);

        }
        else
        {
            pnl_vect_set(V,j,(-pow((1+pow(omegaj,2)), -1)*(cos((-a)*omegaj)-exp(a)+omegaj*s
        }
    }
}

```

```

static double Valgt(double thetav,double T,double a_global,double b_global,doubl
{
    PnlVect *gtt;
    PnlVect *gt;
    double result;
    double Vj;

    int j;
    gtt = pnl_vect_create(N);
    gt = pnl_vect_create(N);

    Valcf(thetav,T,a_global,b_global,r,N,sigmav,kappav,rho,x,fee,q,v0,omega,cf);

    Valvt(a_global,b_global,alphav,P,M,omega,V,N,fee,T);
    cf0(cf);

    for (j=0;j<N;j++)
    {
        Vj=pnl_vect_get(V,j);
        pnl_vect_set(gtt,j,exp(-r*T)*Vj*pnl_vect_complex_get_real(cf,j));
    }

    pnl_vect_set(gt,0,pnl_vect_sum(gtt));

    pnl_vect_free(&gtt);

    result = pnl_vect_get(gt,0);
    pnl_vect_free(&gt);
    return result;
}

static double Valft(double r,double x,double P,double alphav,double sigmav,doubl
{
    double ft=0;
    double t=0;
    double D=pow(pow(kappav-rho*sigmav,2),0.5);
    double G=(kappav-rho*sigmav-D)/(kappav-rho*sigmav+D);

```

```

int i;
for(i=0;i<1000;i++)
{
ft +=fee*exp(-r*t)*exp(x)*alphav*P*M*exp(T/1000*(r-fee)+v0/pow(sigmav,2)*(1-ex
t += T/1000;

}
return ft;
}

```

```

static double fee(int maximum_number_of_loop, double tolerance, double T, double
{
double fee1=0.00;
double fee2=0.01;
double valueg1;
double valueg2;
double valuef1;
double valuef2;
double feexx;
double step;

int number_of_loop;

number_of_loop=0;
step=tolerance*2.;
//    tolerance = 0.0000001;

while( (fabs(step) > tolerance) && (number_of_loop < maximum_number_of_loop))
{
number_of_loop++;
valueg1=Valgt(thetav, T, a, b, alphav, P, M, r, N, sigmav, kappav, rho, x, fee
valueg2=Valgt(thetav, T, a, b, alphav, P, M, r, N, sigmav, kappav, rho, x, fee
valuef1=Valft(r, x, P, alphav, sigmav, rho, thetav, kappav, fee1, M, v0, T);
valuef2=Valft(r, x, P, alphav, sigmav, rho, thetav, kappav, fee2, M, v0, T);
step = (valueg1-valuef1)/((valueg1-valueg2)/(fee1-fee2)-(valuef1-valuef2)/(fee
feexx=fee1-step;
fee2=fee1;
fee1=feexx;
}
return feexx;
}

```

```

}

/*Compute Price Option*/
int AP_FourierCosine_GMIB_Hes(double y,double g,double A0, double maturity, dou
{

    double premium;
    PnlVect *V,*omega, *gt;
    PnlVectComplex *cf;
    double M,ZC,x,L,c1,c2,a_global,b_global;

    premium=A0;

    L=20;

    c1=r*maturity+(1-exp(-kappa*maturity))*(theta-v0)/(2.0*kappa)-0.5*theta*maturi
    c2=(1.0/(8.0*pow(kappa,3)))*(sigma*maturity*kappa*exp(-kappa*maturity)*(v0-the
    a_global=c1-L*pow(fabs(c2),0.5);
    b_global=c1+L*pow(fabs(c2),0.5);

    ZC=1/r*(1-1/(pow((1+r),(double)y)));
    M =exp(rollup_rate*maturity)*g*ZC;
    x=log(A0/(alpha*premium*M));

    omega = pnl_vect_create (N);
    V= pnl_vect_create (N);
    cf =pnl_vect_complex_create (N);
    gt = pnl_vect_create(N);

    Valomega(a_global, b_global, N,omega);

    // 10 is the number of loop in the secante method.

    *ptprice = fee(100,0.000001,maturity,x,a_global,b_global,alpha,theta,premium,M

    pnl_vect_free(&omega);
    pnl_vect_free(&V);
    pnl_vect_complex_free(&cf);
    pnl_vect_free(&gt);

    return OK;

```

```
}
```

```
int CALC(AP_FOURIERCOSINE_GMIB_HES)(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 AP_FourierCosine_GMIB_Hes(ptOpt->TermCertainAnnuitiyMaturity.Val.V_DATE
                                     ptOpt->Maturity.Val.V_DATE - ptMod->T.Val.V_D
                                     ptMod->MeanReversion.Val.V_PDOUBLE,
                                     ptMod->LongRunVariance.Val.V_PDOUBLE,
                                     ptMod->Sigma.Val.V_PDOUBLE,
                                     ptMod->Rho.Val.V_PDOUBLE,
                                     Met->Par[0].Val.V_PINT,
                                     &(Met->Res[0].Val.V_DOUBLE));
}
```

```
static int CHK_OPT(AP_FOURIERCOSINE_GMIB_HES)(void *Opt, void *Mod)
{
    if ((strcmp(((Option *)Opt)->Name, "GMIB") == 0))
        return OK;
    else
        return WRONG;
}
#endif //PremiaCurrentVersion
```

```
static int MET(Init)(PricingMethod *Met, Option *Opt)
{
    if (Met->init == 0)
    {
        Met->init = 1;
        Met->Par[0].Val.V_INT = 256;
    }

    return OK;
}
```

```

PricingMethod MET(AP_FOURIERCOSINE_GMIB_HES) =
{
    "AP_FOURIERCOSINE_GMIB_HES",
    {
        {"SpaceStepNumber", INT2, {100}, ALLOW},
        {" ", PREMIA_NULLTYPE, {0}, FORBID}
    },
    CALC(AP_FOURIERCOSINE_GMIB_HES),
    { {"Fair Fee", DOUBLE, {100}, FORBID},
      {" ", PREMIA_NULLTYPE, {0}, FORBID}
    },
    CHK_OPT(AP_FOURIERCOSINE_GMIB_HES),
    CHK_ok,
    MET(Init)
};

```