

[Help](#)

```
#include "
href../../../../mod/bs1d/bs1d_stda/bs1d_stda_h_src.pdfbs1d_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_GMDB_BS)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
int CALC(AP_FOURIERCOSINE_GMDB_BS)(void *Opt, void *Mod, PricingMethod *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
#else

static void Valomega(double a_global, double b_global, int N, PnlVect *omega)
{
    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 t, double T,double a_global,double b_global,double r,i
{
    double omegaj,a;
    int j;
    a = a_global - fee*t;
    for (j=0;j<N;j++)
    {
        omegaj=pnl_vect_get(omega,j);
        pnl_vect_complex_set(cf,j,Cexp(CRsub(Cadd(Complex(0,((r-0.5*pow(sigmav,2))*T/1
    }
}
```

```

static void cf0(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, double M)
{
    int j;
    double omegaj,a,b;
    a = a_global - fee*T;
    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 d, double A0, double rollup_rate, double T,double alp
{
    PnlVect *gtt;
    double Vi,gt,t,result,L,c1,c2,x,M,a_global,b_global;
    int i,j;

    L=20;
    result = 0;
    t=T/1000;

    gtt = pnl_vect_create(N);

```

```

for (j=0;j<1000;j++)
{

c1=(r-0.5*pow(sigmav,2))*t;

c2=pow(sigmav,2)*t;

a_global=c1-L*pow(fabs(c2),0.5);
b_global=c1+L*pow(fabs(c2),0.5);

M=exp(rollup_rate*t);

x=log(A0/(alphav*P*M));

Valomega(a_global, b_global, N,omega);

Valcf(t, T, a_global,b_global,r,N,sigmav,x,fee,q,omega,cf);

cf0(cf);

Valvt(a_global,b_global,alphav,P,M,omega,V,N,fee,t);

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

Vi=pnl_vect_get(V,i);

pnl_vect_set(gtt,i,exp(-r*t)*Vi*pnl_vect_complex_get_real(cf,i));
}

gt=pnl_vect_sum(gtt);
result += gt * d * exp(-d*t)* T/1000;
t+= T/1000;
}
pnl_vect_free(&gtt);
return result;
}

```

```

static double Valft(double A0, double d, double r,double sigmav,double fee,doubl
{

```

```

double ft;

ft=(d*fee*A0*exp(T/1000*(r-fee)-0.5*pow(sigmav,2))/(-d-r))*(exp((-d-r)*T)-1);

return ft;

}

static double fee(double d,double rollup_rate, double A0, int maximum_number_of_
{
    double fee1=0.00;
    double fee2=0.1;
    double valueg1;
    double valueg2;
    double valuef1;
    double valuef2;
    double feexx;
    double step;
    int number_of_loop;

    number_of_loop=0;
    step=tolerance*2.;

    while( (fabs(step) > tolerance) && (number_of_loop < maximum_number_of_loop))
    {
        number_of_loop++;
        valueg1=Valgt(d,A0,rollup_rate,T, alphav, P, r, N, sigmav, fee1, q, omega, V,
        valueg2=Valgt(d,A0,rollup_rate,T, alphav, P, r, N, sigmav, fee2, q, omega, V,
        valuef1=Valft(A0,d,r, sigmav, fee1, T);
        valuef2=Valft(A0,d,r, sigmav, fee2, 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_GMDB_BS(double A0, double maturity, double r, double divid

```

```

{
    PnlVect *V,*omega;
    PnlVectComplex *cf;
    double premium;

    premium = A0;

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

    *ptprice = fee(d,rollup_rate,A0,100,0.00000001,maturity,alpha,premium,r,N,sigm

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

    return OK;
}

int CALC(AP_FOURIERCOSINE_GMDB_BS)(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_GMDB_BS(ptMod->S0.Val.V_PDOUBLE,ptOpt->Maturity.Val.V_

}

static int CHK_OPT(AP_FOURIERCOSINE_GMDB_BS)(void *Opt, void *Mod)
{
    if ((strcmp(((Option *)Opt)->Name, "GMDB") == 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_GMDB_BS) =
{
    "AP_FOURIERCOSINE_GMDB_BS",
    {
        {"SpaceStepNumber", INT2, {100}, ALLOW},
        {" ", PREMIA_NULLTYPE, {0}, FORBID}
    },
    CALC(AP_FOURIERCOSINE_GMDB_BS),
    { {"Fair Fee", DOUBLE, {100}, FORBID},
      {" ", PREMIA_NULLTYPE, {0}, FORBID}
    },
    CHK_OPT(AP_FOURIERCOSINE_GMDB_BS),
    CHK_ok,
    MET(Init)
};

```