

[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_GMIB_BS)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
int CALC(AP_FOURIERCOSINE_GMIB_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 a_global,double b_global,double r,int N, doub
{
    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-f
    }
}
```

```

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*sigmav)));
        }
    }
}

static double Valgt(double T,double a_global,double b_global,double alphav, double M)
{
    PnlVect *gtt;
    double Vi;
    double gt;
    int i;

    gtt = pnl_vect_create(N);

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

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

```

```

    cf0(cf);

    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);
    pnl_vect_free(&gtt);

    return gt;
}

static double Valft(double r,double x,double P,double alphav,double sigmav,doubl
{
    double ft=0;
    double t=0;
    int i;
    for(i=0;i<1000;i++)
    {
        ft +=fee*exp(-r*t)*exp(x)*alphav*P*M*exp(T/1000*(r-fee)-0.5*pow(sigmav,2))*T/1
        t += T/1000;
    }
    return ft;
}

static double fee(int maximum_number_of_loop, double tolerance, double T ,double
{
    double fee1,fee2,valuef1,valuef2,valueg1,valueg2,step,feexx;
    int number_of_loop;
    fee1=0.00;
    fee2=0.1;
    number_of_loop=0;
    step=tolerance*2.;

    while( (fabs(step) > tolerance) && (number_of_loop < maximum_number_of_loop))
    {
        number_of_loop++;
        valueg1=Valgt(T, a, b, alphav, P, M, r, N, sigmav, x, fee1, q, omega, V, cf);
        valueg2=Valgt(T, a, b, alphav, P, M, r, N, sigmav, x, fee2, q, omega, V, cf);
        valuef1=Valft(r, x, P, alphav, sigmav, fee1, M, T);

```

```

    valuef2=Valft(r, x, P, alphav, sigmav, fee2, M, T);
    step = (valueg1-valuef1)/((valueg1-valueg2)/(fee1-fee2)-(valuef1-valuef2)/(fee1-fee2));
    feexx=fee1-step;
    fee2=fee1;
    fee1=feexx;
    }
    return feexx;
}

/*Compute Price Option*/
int AP_FourierCosine_GMIB_BS(double y,double g,double A0, double maturity, double r, double N)
{
    double premium;
    PnlVect *V,*omega;
    PnlVectComplex *cf;
    double M,x,L,c1,c2,a_global,b_global;
    double ZC;

    premium = A0;
    L=20;
    c1=(r-0.5*pow(sigma,2))*maturity;
    c2=pow(sigma,2)*maturity;
    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),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);

    Valomega(a_global, b_global, N,omega);
    // 10 is the number of loop in the secante method.
    *ptprice = fee(100,0.00000001,maturity,x,a_global,b_global,alpha,premium,M,r,N);

    pnl_vect_free(&omega);
    pnl_vect_free(&V);
}

```

```

    pnl_vect_complex_free(&cf);

    return OK;
}

int CALC(AP_FOURIERCOSINE_GMIB_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_GMIB_BS(ptOpt->TermCertainAnnuitiyMaturity.Val.V_DATE,
                                     ptOpt->Maturity.Val.V_DATE - ptMod->T.Val.V_DATE,
                                     Met->Par[0].Val.V_PINT,
                                     &(Met->Res[0].Val.V_DOUBLE));
}

static int CHK_OPT(AP_FOURIERCOSINE_GMIB_BS)(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_BS) =

```

```

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

```