

[Help](#)

```

#include "cev1d_std.h"
#include "pnl/pnl_cdf.h"
#include "pnl/pnl_finance.h"
#include "pnl/pnl_root.h"
#include "pnl/pnl_cdf.h"
#include "pnl/pnl_finance.h"
#include "pnl/pnl_root.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2013+2) //The "#els
static int CHK_OPT(AP_BGM_Cev)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
int CALC(AP_BGM_Cev)(void *Opt, void *Mod, PricingMethod *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
#else

static int ApBGM_Cev(double S0, NumFunc_1 *p, double T, double r, double q, double
                    double beta, double *ATM_price, double *AS_price)
{
    double K;
    double sigma2_black_T, tilde_sigma2_black_T, A, B, d_1, d_2, tilde_d_1, tilde_

    K = p->Par[0].Val.V_PDOUBLE;

    x0 = log(S0);
    mu = 2 * (beta - 1) * (r - q);

    if (r == q)
    {
        sigma2_black_T = T * SQR(v) * exp(2 * (beta - 1) * x0);
        alpha1_T = (beta - 1) * SQR(v) * SQR(v) * exp(4 * (beta - 1) * x0) * SQR(T

        tilde_sigma2_black_T = T * SQR(v) * pow(K, 2 * (beta - 1));
        tilde_alpha1_T = (beta - 1) * SQR(v) * SQR(v) * pow(K, 4 * (beta - 1)) * S
    }
    else

```

```

{
    sigma2_black_T = (exp(mu * T) - 1) * SQR(v) * exp(2 * (beta - 1) * x0) / (
    alpha1_T = (beta - 1) * SQR(v) * SQR(v) * exp(4 * (beta - 1) * x0) * (0.5

    tilde_sigma2_black_T = SQR(v) * pow(K, 2 * (beta - 1)) * (exp(-mu * T) - 1
    tilde_alpha1_T = (beta - 1) * SQR(v) * SQR(v) * pow(K, 4 * (beta - 1)) * (

}

d_1 = (1 / sqrt(sigma2_black_T)) * (log(S0 * exp(-q * T) / (K * exp(-r * T)))
d_2 = d_1 - sqrt(sigma2_black_T);

tilde_d_1 = (1 / sqrt(tilde_sigma2_black_T)) * (log(S0 * exp(-q * T) / (K * ex
tilde_d_2 = tilde_d_1 - sqrt(tilde_sigma2_black_T);

A = S0 * exp(-q * T) * cdf_nor(d_1) - K * exp(-r * T) * cdf_nor(d_2);
B = S0 * exp(-q * T) * cdf_nor(tilde_d_1) - K * exp(-r * T) * cdf_nor(tilde_d_

Greek_1 = exp(-q * T) * pnl_normal_density(d_1) / (S0 * sqrt(sigma2_black_T));

Greek_2 = -(Greek_1 / S0) * (1 + d_1 / sqrt(sigma2_black_T));

Greek_3 = exp(-r * T) * pnl_normal_density(tilde_d_2) / (K * sqrt(tilde_sigma2

Greek_4 = -(Greek_3 / K) * (1 - d_2 / sqrt(tilde_sigma2_black_T));

//At the Money Call Price
*ATM_price = A + alpha1_T * (1.5 * SQR(S0) * Greek_1 + CUB(S0) * Greek_2);
//Second order approximationfor call options based on the ATM (AT the Money) loc

//At the Strike CallPrice
*AS_price = B + tilde_alpha1_T * (1.5 * SQR(K) * Greek_3 + CUB(K) * Greek_4);;

//Put case by parity
if ((p->Compute) == &Put)
{
    *ATM_price = *ATM_price - S0 + K * exp(-r * T);
    *AS_price = *ATM_price - S0 + K * exp(-r * T);
}
return OK;
}

```

```

int CALC(AP_BGM_Cev)(void *Opt, void *Mod, PricingMethod *Met)
{
    TYPEOPT *ptOpt = (TYPEOPT *)Opt;
    TYPEMOD *ptMod = (TYPEMOD *)Mod;
    int status;
    double r, divid;

    r = log(1. + ptMod->R.Val.V_DOUBLE / 100.);
    divid = log(1. + ptMod->Divid.Val.V_DOUBLE / 100.);

    status = ApBGM_Cev(ptMod->S0.Val.V_PDOUBLE,
                       ptOpt->PayOff.Val.V_NUMFUNC_1,
                       ptOpt->Maturity.Val.V_DATE - ptMod->T.Val.V_DATE,
                       r,
                       divid,
                       ptMod->v.Val.V_PDOUBLE,
                       ptMod->beta.Val.V_PDOUBLE,
                       &(Met->Res[0].Val.V_DOUBLE), &(Met->Res[1].Val.V_DOUBLE));

    return status;
}

static int CHK_OPT(AP_BGM_Cev)(void *Opt, void *Mod)
{
    if ((strcmp(((Option *)Opt)->Name, "CallEuro") == 0) || (strcmp(((Option *)Opt)
        return OK;
    return WRONG;
}

#endif //PremiaCurrentVersion
static int MET(Init)(PricingMethod *Met, Option *Opt)
{
    if (Met->init == 0) Met->init = 1;
    return OK;
}

PricingMethod MET(AP_BGM_Cev) =
{
    "AP_BGM_Cev",
    {" ", PREMIA_NULLTYPE, {0}, FORBID}},

```

```
CALC(AP_BGM_Cev),  
{ {"At the Money Price", DOUBLE, {100}, FORBID}, {"At the Strike Price", DOUBL  
  {" ", PREMIA_NULLTYPE, {0}, FORBID}  
},  
CHK_OPT(AP_BGM_Cev),  
CHK_ok,  
MET(Init)  
};
```