

[Help](#)

```
#include <stdlib.h>
#include "bs1d_std.h"
#include "error_msg.h"

static int ThirdMoment(int am, double s, NumFunc_1 *payoff, double t, double r,
{
    double h, u, d, scan, p, q, lowerstock, iv, stock, Q, R;
    double *P;
    int i, j;

    /*Price array*/
    P = malloc((N + 1) * sizeof(double));
    if (P == NULL)
        return MEMORY_ALLOCATION_FAILURE;

    /*Up and Down factors*/
    h = t / (double)N;
    Q = exp(sigma * sigma * h);
    R = exp((r - divid) * h);

    u = R * Q * (1.0 + Q + sqrt(Q * Q + 2.0 * Q - 3.0)) / 2.0;
    d = R * Q * (1.0 + Q - sqrt(Q * Q + 2.0 * Q - 3.0)) / 2.0;

    scan = u / d;

    /*Discounted Risk-Neutral Probability*/
    p = (R - d) / (u - d);
    q = 1.0 - p;
    p *= exp(-r * h);
    q *= exp(-r * h);

    /*Terminal Values*/
    lowerstock = s;
    for (i = 0; i < N; i++)
        lowerstock *= d;

    stock = lowerstock;
    for (i = 0; i <= N; i++)
    {
```

```

        iv = (payoff->Compute)(payoff->Par, stock);
        P[i] = iv;
        stock *= scan;
    }

/*Backward Resolution*/
for (i = N; i > 1; i--)
{
    lowerstock /= d;
    stock = lowerstock;
    for (j = 0; j < i; j++)
    {
        P[j] = q * P[j] + p * P[j + 1];
        if (am)
        {
            iv = (payoff->Compute)(payoff->Par, stock);
            P[j] = MAX(iv, P[j]);
        }
        stock *= scan;
    }

}

lowerstock /= d;
stock = lowerstock;

/*Delta*/
*ptdelta = (P[1] - P[0]) / (stock * u - stock * d);

/*First time step*/
P[0] = q * P[0] + p * P[1];
if (am)
{
    iv = (payoff->Compute)(payoff->Par, stock);
    P[0] = MAX(iv, P[0]);
}

/*Price*/
*ptprice = P[0];

/*Memory desallocation*/

```

```

    free(P);

    return OK;
}

static int CHK_OPT(TR_ThirdMoment)(void *Opt, void *Mod)
{
    return OK;
}

int CALC(TR_ThirdMoment)(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 ThirdMoment(ptOpt->EuOrAm.Val.V_BOOL, ptMod->S0.Val.V_PDOUBLE, ptOpt->P
        ptOpt->Maturity.Val.V_DATE - ptMod->T.Val.V_DATE, r, divid,
        Met->Par[0].Val.V_INT, &(Met->Res[0].Val.V_DOUBLE), &(Met->
    }

static int MET(Init)(PricingMethod *Met, Option *Opt)
{
    if (Met->init == 0)
    {
        Met->init = 1;

        Met->Par[0].Val.V_INT2 = 100;

    }

    return OK;
}

PricingMethod MET(TR_ThirdMoment) =
{
    "TR_ThirdMoment",

```

```
{{"StepNumber", INT2, {100}, ALLOW}, {" ", PREMIA_NULLTYPE, {0}, FORBID}},  
CALC(TR_ThirdMoment),  
{{"Price", DOUBLE, {100}, FORBID}, {"Delta", DOUBLE, {100}, FORBID} , {" ", PR  
CHK_OPT(TR_ThirdMoment),  
CHK_tree,  
MET(Init)  
};
```