

[Help](#)

```
#include "
href../../../../mod/bs1d/bs1d_limdisc/bs1d_limdisc_h_src.pdfbs1d_limdisc.h"
#include "pnl/pnl_cdf.h"

int MOD_OPT(ChkMix)(Option *Opt, Model *Mod)
{
    TYPEOPT *ptOpt = (TYPEOPT *) (Opt->TypeOpt);
    TYPEMOD *ptMod = (TYPEMOD *) (Mod->TypeModel);
    int status = OK;

    if (ptOpt->Maturity.Val.V_DATE <= ptMod->T.Val.V_DATE)
    {
        Fprintf(TOSCREENANDFILE, "Current date greater than maturity!\ n");
        status += 1;
    };

    if ((ptOpt->Limit.Val.V_NUMFUNC_1)->Par[0].Val.V_DATE < ptMod->T.Val.V_DATE)
    {
        Fprintf(TOSCREENANDFILE, "Current date upper than Starting date!\ n");
        status += 1;
    };

    if ((ptOpt->Limit.Val.V_NUMFUNC_1)->Par[0].Val.V_DATE > ptOpt->Maturity.Val.V_
    {
        Fprintf(TOSCREENANDFILE, "Maturity lower than Starting date!\ n");
        status += 1;
    };
    return status;
}

extern PricingMethod MET(AP_BroadieGlassermanKou);
extern PricingMethod MET(AP_FusaiAbrahamsSgarra);
extern PricingMethod MET(MC_VarianceReduction);
extern PricingMethod MET(TR_CK);
extern PricingMethod MET(FD_LimDisc);

PricingMethod *MOD_OPT(methods)[] =
{
    &MET(AP_BroadieGlassermanKou),
```

```

    &MET(AP_FusaiAbrahamsSgarra),
    &MET(MC_VarianceReduction),
    &MET(TR_CK),
    &MET(FD_LimDisc),
    NULL
};

extern DynamicTest MOD_OPT(test);
DynamicTest *MOD_OPT(tests)[] =
{
    &MOD_OPT(test),
    NULL
};

Pricing MOD_OPT(pricing) =
{
    ID_MOD_OPT,
    MOD_OPT(methods),
    MOD_OPT(tests),
    MOD_OPT(ChkMix)
};

/* shared utility function.
 *
 */

int MOD_OPT(formula_lib)(double s, double k, double r, double divid, double sigma,
                        double *A, double *B, double *C, double *D, double *E,
{
    double b, x1, x2, y1, y2, z, mu, lambda, sigmasqrt;

    sigmasqrt = sigma * sqrt(t);
    b = r - divid;
    mu = (b - SQR(sigma) / 2.) / SQR(sigma);
    lambda = sqrt(SQR(mu) + 2.*r / SQR(sigma));
    x1 = log(s / k) / sigmasqrt + (1 + mu) * sigmasqrt;
    x2 = log(s / l) / sigmasqrt + (1 + mu) * sigmasqrt;
    y1 = log(SQR(l) / (s * k)) / sigmasqrt + (1 + mu) * sigmasqrt;
    y2 = log(l / s) / sigmasqrt + (1 + mu) * sigmasqrt;
    z = log(l / s) / sigmasqrt + lambda * sigmasqrt;
    *A = phi * s * exp((b - r) * t) * cdf_nor(phi * x1) - phi * k * exp(-r * t) *

```

```

*B = phi * s * exp((b - r) * t) * cdf_nor(phi * x2) - phi * k * exp(-r * t) *
*C = phi * s * exp((b - r) * t) * pow(1 / s, 2.*(1. + mu)) * cdf_nor(eta * y1)
      phi * k * exp(-r * t) * pow(1 / s, 2.*mu) * cdf_nor(eta * y1 - eta * sigma)
*D = phi * s * exp((b - r) * t) * pow(1 / s, 2.*(1. + mu)) * cdf_nor(eta * y2)
      phi * k * exp(-r * t) * pow(1 / s, 2.*mu) * cdf_nor(eta * y2 - eta * sigma)
*E = rebate * exp(-r * t) * (cdf_nor(eta * x2 - eta * sigmasqrt) - pow(1 / s,
*F = rebate * (pow(1 / s, mu + lambda) * cdf_nor(eta * z) + pow(1 / s, mu - lambda)

return OK;
}

```