

## [Help](#)

```
#include "
href../../../../mod/dps/dps_std/dps_std_h_src.pdf\dps_std.h"
#include "
href../../../../common/math/equity_pricer/levy_diffusion_h_src.pdf\math/equity_price
#include "
href../../../../common/math/equity_pricer/carr_h_src.pdf\math/equity_pricer/carr.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2010+2) //The "#els
static int CHK_OPT(CF_AttariDPS)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
int CALC(CF_AttariDPS)(void *Opt, void *Mod, PricingMethod *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
#else

int CALC(CF_AttariDPS)(void *Opt, void *Mod, PricingMethod *Met)
{
    TYPEOPT *ptOpt = (TYPEOPT *)Opt;
    TYPEMOD *ptMod = (TYPEMOD *)Mod;
    NumFunc_1 *p;
    double jump_drift;
    int option_type;
    int std = 1;
    Option_Eqd *op;
    DPS_diffusion *Process = DPS_diffusion_create(ptMod->Eta.Val.V_PDOUBLE,
                                                    ptMod->Kappa.Val.V_PDOUBLE,
                                                    ptMod->Rho.Val.V_PDOUBLE,
                                                    ptMod->Theta.Val.V_PDOUBLE,
                                                    ptMod->Sigma0.Val.V_PDOUBLE,
                                                    ptMod->MeanS.Val.V_PDOUBLE,
                                                    ptMod->SigmaS.Val.V_PDOUBLE,
                                                    ptMod->LambdaS.Val.V_PDOUBLE,
                                                    ptMod->MeanV.Val.V_PDOUBLE,
                                                    ptMod->LambdaV.Val.V_PDOUBLE,
                                                    ptMod->MeanSV.Val.V_PDOUBLE,
```

```

        ptMod->SigmaSV.Val.V_PDOUBLE,
        ptMod->MeanVS.Val.V_PDOUBLE,
        ptMod->LambdaSV.Val.V_PDOUBLE,
        ptMod->RhoSV.Val.V_PDOUBLE,
        &jump_drift);
Levy_diffusion *Levy = Levy_diffusion_create(Process, &DPS_diffusion_character
p = ptOpt->PayOff.Val.V_NUMFUNC_1;
if ((p->Compute) == &Call)
    option_type = 1;
else if ((p->Compute) == &Put)
    option_type = 2;
else
    option_type = 3;

op = option_eqd_create(ptOpt->EuOrAm.Val.V_BOOL, option_type, std, ptMod->S0.V
option_eqd_set_rate(op, log(1. + ptMod->R.Val.V_DOUBLE / 100.), log(1. + ptMod

AttariMethod_Vanilla_option_LD(op, 0.1, Levy);
(Met->Res[0].Val.V_DOUBLE) = op->price;
(Met->Res[1].Val.V_DOUBLE) = op->delta;
free(op);
free(Levy);
free(Process);
return OK;
}

static int CHK_OPT(CF_AttariDPS)(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(CF_AttariDPS) =
{
    "CF_Attari_DPS",
    {{" ", PREMIA_NULLTYPE, {0}, FORBID}},
    CALC(CF_AttariDPS),
    { {"Price", DOUBLE, {100}, FORBID},
      {"Delta", DOUBLE, {100}, FORBID} ,
      {" ", PREMIA_NULLTYPE, {0}, FORBID}
    },
    CHK_OPT(CF_AttariDPS),
    CHK_ok,
    MET(Init)
};

```