

[Help](#)

```
#include "schwartz_std.h"
#include "pnl/pnl_mathtools.h"
#include "pnl/pnl_vector.h"
#include "pnl/pnl_matrix.h"
#include "pnl/pnl_cdf.h"

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

static int ap_schwartz(double r, double divid, double sigma_d, double sigma_s, d
{
    double *b, *t_i, prefact, bla, sum_b, sum_bf, psi, sum_bs1, sum_b2s2, d1, d2;
    int i, j, n;

    n = (int)((swap_mat - opt_mat) / tenor);
    t_i = (double *)malloc(n * sizeof(double));

    prefact = sigma_d / alpha;
    bla = 2 * sigma_s * rho;
    sum_b = 0;
    sum_bf = 0;
    psi = 0;
    sum_bs1 = 0;
    sum_b2s2 = 0;

    b = (double *)malloc(n * sizeof(double));

    for (i = 0; i < n; i++)
    {

        t_i[i] = opt_mat + (i + 1) * tenor;
```

```

    b[i] = exp(-r * t_i[i]);

    sum_b += b[i];

    sum_bf += exp(-divid * t_i[i]);

    sum_bs1 += b[i] * (exp(-alpha * (t_i[i] - tenor)) - exp(-alpha * t_i[i]));
}

// The big sum to compute Psi
for (i = 0; i < n; i++)
{
    for (j = 0; j < n; j++)
    {
        sum_b2s2 += b[i] * b[j] * (exp(-alpha * (t_i[i] + t_i[j] - 2 * tenor)));
    }
}

psi = tenor * (prefact * prefact + sigma_s * sigma_s - bla * prefact) +
    prefact * (sum_bs1 * (-2 * prefact + bla) + prefact / sum_b * sum_b2s2 /

psi = sqrt(psi);

d1 = -log(K) / psi + psi / 2;
d2 = d1 - psi;

*ptprice = sum_bf * Nominal * (pnl_cdfnor(d1) - K * pnl_cdfnor(d2));

free(b);
free(t_i);

return OK;
}

int CALC(AP_SCHWARTZ)(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_schwartz(r, divid, ptMod->sigmad.Val.V_PDOUBLE, ptMod->sigmas.Val.V_
        ptOpt->Nominal.Val.V_PDOUBLE,
        ptOpt->ResetPeriod.Val.V_DATE,
        ptOpt->OMaturity.Val.V_DATE - ptMod->T.Val.V_DATE - ptMod->
        ptOpt->BMaturity.Val.V_DATE - ptMod->T.Val.V_DATE - ptMod->
        ptOpt->FixedRate.Val.V_PDOUBLE,
        &(Met->Res[0].Val.V_DOUBLE)
    );
}

static int CHK_OPT(AP_SCHWARTZ)(void *Opt, void *Mod)
{
    if ((strcmp(((Option *)Opt)->Name, "PayerSwaption") == 0))
        return OK;

    return WRONG;
}

#endif //PremiaCurrentVersion

static int MET(Init)(PricingMethod *Met, Option *Opt)
{
    //int type_generator;
    if (Met->init == 0)
    {
        Met->init = 1;
        Met->HelpFilenameHint = "ap_schwartz_swaption";
    }

    return OK;
}

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

```

```
CALC(AP_SCHWARTZ),  
{ {"Price", DOUBLE, {100}, FORBID},  
  {" ", PREMIA_NULLTYPE, {0}, FORBID}  
},  
CHK_OPT(AP_SCHWARTZ),  
CHK_ok,  
MET(Init)  
};
```