

[Help](#)

```

#include <stdlib.h>
#include "vasicek1d_std.h"

/*Product Variables*/
static double A, B;
static int nb_payement;
static double *C;

/*Zero Coupon Bond*/
static double zcb_vasicek1d(double ti, double Ti, double r, double theta, double sigma)
{
    B = (1. / k) * (1. - exp(-k * (Ti - ti)));
    A = exp((theta - SQR(sigma) / (2.*SQR(k))) * (B - Ti + ti) - (SQR(sigma) / (4. * k)));

    return A * exp(-B * r);
}

/*Computation of Critical Rate*/
static double phi(double rr, double date, double periodicity, double first_payement)
{
    int i;
    double sum, sum_der, ti;

    sum = 0.;
    sum_der = 0.;
    for (i = 0; i <= nb_payement; i++)
    {
        ti = first_payement + (double)i * periodicity;
        sum += C[i] * zcb_vasicek1d(option_maturity, ti, rr, theta, k, sigma);
        sum_der += C[i] * zcb_vasicek1d(option_maturity, ti, rr, theta, k, sigma) * ti;
    }
    return (sum - 1.) / sum_der;
}

static double Critical_Rate(double date, double periodicity, double first_payement)
{
    const double precision = 0.0001;

```

```

double previous, current = 0.;
do
{
    previous = current;
    current = current - phi(current, date, periodicity, first_payement, option
}
while (!(fabs((previous - current)) <= precision));
return current;
}

/*Put Option on Zero Coupon Bond*/
static double zbp_vasicek1d(double t, double T, double S, double new_K, double r
{
    double PtS, PtT;
    double d1, d2, sigma_p;

    PtT = zcb_vasicek1d(t, T, r, theta, k, sigma);
    PtS = zcb_vasicek1d(t, S, r, theta, k, sigma);
    sigma_p = sigma * sqrt((1. - exp(-2.*k * (T - t))) / (2 * k)) * (1. / k) * (1.
    d1 = 1. / (sigma_p) * log(PtS / (PtT * new_K)) + 0.5 * sigma_p;
    d2 = d1 - sigma_p;

    return new_K * PtT * cdf_nor(-d2) - PtS * cdf_nor(-d1);
}

/*Payer Swaption*/
static int ps_vasicek1d(double r, double k, double date, double sigma, double th
{
    double sum, ti, new_K, critical_r, first_payement;
    int i;

    first_payement = option_maturity + periodicity;
    nb_payement = (int)((contract_maturity - first_payement) / periodicity);

    /*Payer Swaption=Put Option on Coupon-Bearing Bond*/
    C = malloc((nb_payement + 1) * sizeof(double));
    for (i = 0; i <= nb_payement; i++)
    {

        if (i != nb_payement)
            C[i] = K * periodicity;

```

```

        else
            C[i] = (1. + K * periodicity);

    }

    /*Jamshidian decomposition*/
    /*Computation of critical rate*/
    critical_r = Critical_Rate(date, periodicity, first_payement, option_maturity,

    /*Portfolio of PUT Option*/
    sum = 0.;
    for (i = 0; i <= nb_payement; i++)
    {
        ti = first_payement + (double)i * periodicity;
        /*Strike*/
        new_K = zcb_vasicek1d(option_maturity, ti, critical_r, theta, k, sigma);
        sum += C[i] * zbp_vasicek1d(date, option_maturity, ti, new_K, r, theta, k,

    }

    /*Price*/
    *price = Nominal * sum;

    free(C);
    return OK;
}

int CALC(CF_PayerSwaption)(void *Opt, void *Mod, PricingMethod *Met)
{
    TYPEOPT *ptOpt = (TYPEOPT *)Opt;
    TYPEMOD *ptMod = (TYPEMOD *)Mod;

    return ps_vasicek1d(ptMod->r0.Val.V_PDOUBLE, ptMod->k.Val.V_DOUBLE, ptMod->T.Val.V_PDOUBLE,
        ptMod->theta.Val.V_PDOUBLE, ptOpt->Nominal.Val.V_PDOUBLE,
        ptOpt->OMaturity.Val.V_DATE, ptOpt->BMaturity.Val.V_DATE,

}

static int CHK_OPT(CF_PayerSwaption)(void *Opt, void *Mod)
{
    return strcmp(((Option *)Opt)->Name, "PayerSwaption");
}

```

```
}

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

    return OK;
}

PricingMethod MET(CF_PayerSwaption) =
{
    "CF_Vasicek1d_PayerSwaption",
    {" ", PREMIA_NULLTYPE, {0}, FORBID}},
    CALC(CF_PayerSwaption),
    {"Price", DOUBLE, {100}, FORBID}, {" ", PREMIA_NULLTYPE, {0}, FORBID}},
    CHK_OPT(CF_PayerSwaption),
    CHK_ok,
    MET(Init)
} ;
```