

[Help](#)

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

/*Product*/
static double dt, dr, r_min, r_max;
static double *r_vect, *disc, * *Ps;
static double *pu, *pm, *pd;
static long Ns;

/*Memory Allocation*/
static void memory_allocation(long Nt)
{
    int i;
    if ((r_vect = malloc(sizeof(double) * (Ns + 1))) == NULL)
    {
        printf("Allocation error");
        exit(1);
    }
    if ((disc = malloc(sizeof(double) * (Ns + 1))) == NULL)
    {
        printf("Allocation error");
        exit(1);
    }
    if ((pu = malloc(sizeof(double) * (Ns + 1))) == NULL)
    {
        printf("Allocation error");
        exit(1);
    }
    if ((pm = malloc(sizeof(double) * (Ns + 1))) == NULL)
    {
        printf("Allocation error");
        exit(1);
    }
    if ((pd = malloc(sizeof(double) * (Ns + 1))) == NULL)
    {
        printf("Allocation error");
        exit(1);
    }
    if ((Ps = malloc(sizeof(double *) * (Nt + 1))) == NULL)
```

```
{
    printf("Allocation error");
    exit(1);
}
for (i = 0; i <= Nt; i++)
{
    Ps[i] = malloc(sizeof(double) * (Ns + 1));
}

return;
}

/*Memory Desallocation*/
static void free_memory(long Nt)
{
    int i;

    free(r_vect);
    free(pu);
    free(pm);
    free(pd);
    free(disc);

    for (i = 0; i < Nt + 1; i++)
        free(Ps[i]);
    free(Ps);

    return;
}

/*Computation of probabilities*/
static int init_prob(double k, double sigma, double theta, double T, double t0,
{
    double df;
    int j;

    dt = (T - t0) / (double)Nt;
    dr = sigma * sqrt(3.*dt);
    r_min = theta - dr / (2.*k * dt);
```

```

r_max = theta + dr / (2.*k * dt);

Ns = (int)ceil((r_max - r_min) / dr);

memory_allocation(Nt);
for (j = 0; j <= Ns; j++)
{
    r_vect[j] = r_min + (double)j * dr;
    df = k * (theta - r_vect[j]) * dt / dr;
    if (j == 0)
    {
        pu[j] = 1. / 6. + (SQR(df) - df) / 2.;
        pm[j] = df - 2.*pu[j];
        pd[j] = 1. - pu[j] - pm[j];
    }
    else if (j == Ns)
    {
        pd[j] = 1. / 6. + (SQR(df) + df) / 2.;
        pm[j] = -df - 2.*pd[j];
        pu[j] = 1. - pd[j] - pm[j];
    }
    else
    {
        pu[j] = 1. / 6. + (SQR(df) + df) / 2.;
        pd[j] = pu[j] - df;
        pm[j] = 1. - pu[j] - pd[j];
    }
}
return OK;
}

/*Zero Coupon Bond*/
static int zcb_vasicek(long Nt)
{
    int i, j;

    /*Maturity conditions for pure discount Bond*/
    for (j = 0; j <= Ns; j++)
        Ps[Nt][j] = 1.;

    /*Dynamic Programming*/

```

```

for (i = Nt - 1; i >= 0; i--)
    for (j = 0; j <= Ns; j++)
    {
        disc[j] = exp(-r_vect[j] * dt);

        if (j == 0)
            Ps[i][j] = disc[j] * (pu[j] * Ps[i + 1][j + 2] + pm[j] * Ps[i + 1][j + 1] + pd[j] * Ps[i + 1][j]);
        else if (j == Ns)
            Ps[i][j] = disc[j] * (pd[j] * Ps[i + 1][j - 2] + pm[j] * Ps[i + 1][j - 1] + pu[j] * Ps[i + 1][j]);
        else
            Ps[i][j] = disc[j] * (pu[j] * Ps[i + 1][j + 1] + pm[j] * Ps[i + 1][j] + pd[j] * Ps[i + 1][j - 1]);
    }
    return 1.;
}

static int bond_vasicek1d(double r0, double k, double t0, double sigma, double T)
{
    int j;
    double val, val1;

    /*Compute probabilities*/
    init_prob(k, sigma, theta, T, t0, Nt);

    /*Compute Zero Coupon Prices*/
    zcb_vasicek(Nt);

    /*Linear Interpolation*/
    j = 0;
    while (r_vect[j] < r0)
        j++;
    val = Ps[0][j];
    val1 = Ps[0][j - 1];

    /*Price*/
    *price = val + (val - val1) * (r0 - (r_vect[j])) / ((r_vect[j]) - (r_vect[j - 1]));

    /*Memory Disallocation*/
    free_memory(Nt);

    return OK;
}

```

```

}

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

    return bond_vasicek1d(ptMod->r0.Val.V_PDOUBLE, ptMod->k.Val.V_DOUBLE, ptMod->T
        ptMod->theta.Val.V_PDOUBLE, ptOpt->BMaturity.Val.V_DATE,
}

static int CHK_OPT(FD_ZCBond)(void *Opt, void *Mod)
{
    if ((strcmp(((Option *)Opt)->Name, "ZeroCouponBond") == 0))
        return OK;
    else
        return WRONG;
}

static int MET(Init)(PricingMethod *Met, Option *Opt)
{
    if (Met->init == 0)
    {
        Met->init = 1;
        Met->Par[0].Val.V_LONG = 40;
    }
    return OK;
}

PricingMethod MET(FD_ZCBond) =
{
    "FD_Explicit_Vasicek1d_ZCBond",
    { {"TimeStepNumber", LONG, {100}, ALLOW},
      {" ", PREMIA_NULLTYPE, {0}, FORBID}
    },
    CALC(FD_ZCBOND),
    { {"Price", DOUBLE, {100}, FORBID} , {" ", PREMIA_NULLTYPE, {0}, FORBID}},

```

```
    CHK_OPT(FD_ZCBond),  
    CHK_ok,  
    MET(Init)  
} ;
```