

[Help](#)

```
#include <stdlib.h>

#include "math/cdo/cdo.h"
#include "math/cdo/company.h"
#include "copula_stdndc.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
    (2007+2) //The "#else" part of the code will be freely av
    ailable after the (year of creation of this file + 2)
#else

double* get_t_recovery_arg (const VAR *x)
{
    int t_recovery;
    double *arg;
    VAR *Par;
    t_recovery = (x->Val.V_ENUM.value);
    Par = lookup_premia_enum_par(x, t_recovery);
    switch (t_recovery)
    {
        case T_RECOVERY_CONSTANT:
            arg = &(Par[0].Val.V_DOUBLE);
            break;
        case T_RECOVERY_UNIFORM:
            arg = Par[0].Val.V_PNLVECT->array;
            break;
        default:
            arg = NULL;
            break;
    }
    return arg;
}

int      price_cdo(const int      *n_comp,
                   const double *nominal,
                   const int      n_dates,
                   const double *dates,
                   const int      n_tranches,
                   const double *tr,
```

```

        const double *intensity,
        const int     n_rates,
        const double *x_rates,
        const double *y_rates,
        const int     *t_recovery,
        const double *recovery,
        const int     *t_copula,
        const double *p_copula,
        const int     *t_method,
        const int     *p_method,
        double        *price,
        double        *def_leg,
        double        *pay_leg)
{
    double        **tab_sadd;
    double        ***U;
    int           n_sub      = 1;
    double        x_intensity[2];
    double        y_intensity[2];
    int           jt;
    int           jtr;
    int           n_mc;
    int           size_price;
    step_fun      *rates;
    company       **Co;
    CDO           *cdo;
    CDO           *hcdo;
    copula        *cop;
    grid          *x          = NULL;
    grid          *t          = NULL;
    cond_prob     *cp         = NULL;
    double        **numdef    = NULL;
    double        **losses    = NULL;
    grid          **meanloss  = NULL;
    int           jn;
    double        *dl         = NULL;
    double        *pl         = NULL;
    double        *hdl        = NULL;
    double        *hpl        = NULL;
    prod          *produit;

```

```

produit=malloc(sizeof(prod));
produit->nominal=nominal[0];
produit->nb=n_comp[0];
produit->recov=recovery[0];
produit->maturite=n_dates*dates[0];

Co = malloc(*n_comp * sizeof(company*));
x_intensity[0] = 0.;
x_intensity[1] = dates[n_dates-1];
switch (*t_recovery) {
case T_RECOVERY_CONSTANT :
    for (jn = 0; jn < *n_comp; jn++) {
        y_intensity[0] = intensity[jn];
        y_intensity[1] = intensity[jn];
        Co[jn] = init_company_cov_cst(nominal[jn], n_sub, x_
intensity, y_intensity, recovery[0]);
    }
    break;
case T_RECOVERY_UNIFORM :
    for (jn = 0; jn < *n_comp; jn++) {
        y_intensity[0] = intensity[jn];
        y_intensity[1] = intensity[jn];
        Co[jn] = init_company_cov_unif(nominal[jn], n_sub, x_
intensity, y_intensity, recovery[0], recovery[1]);
    }
    break;
}
pnl_rand_init (0, 1, 1);
cdo = init_CDO(*n_comp, Co, n_dates, dates, n_tranches,
tr);
switch (*t_copula) {
case T_COPULA_GAUSS :
    cop = init_gaussian_copula(p_copula[0]);
    break;
case T_COPULA_CLAYTON:
    cop = init_clayton_copula(p_copula[0]);
    break;
case T_COPULA_NIG :
    cop = init_nig_copula(p_copula[0], p_copula[1], p_ copula[2]);
    break;

```

```

case T_COPULA_STUDENT:
    cop= init_student_copula( p_copula[0],p_copula[1]);
    break;
case T_COPULA_DOUBLE_T:
    cop= init_double_t_copula( p_copula[0],p_copula[1],p_copula[2]);
    break;
}
rates = init_cont_linear_sf(n_rates-1, x_rates, y_rates);

produit->rate=compute_sf(rates,1);

switch (*t_method)
{
case T_METHOD_HULL_WHITE_HOMO :
    if(*t_copula != T_COPULA_STUDENT){
        t = init_fine_grid(cdo->dates, p_method[0]);
        cp = init_cond_prob(cdo, cop, t);
        numdef = hw_numdef(cdo, cop, t, cp);
        meanloss = mean_losses_from_numdef(cdo, t, numdef);
        pl = payment_leg(cdo, rates, t, meanloss);
        dl = default_leg(cdo, rates, t, meanloss);
    }
    else{
        t = init_fine_grid(cdo->dates, p_method[0]);
        cp = init_cond_prob(cdo, cop, t);
        numdef = hw_numdef1(cdo, cop, t, cp);
        meanloss = mean_losses_from_numdef(cdo, t, numdef);
        pl = payment_leg(cdo, rates, t, meanloss);
        dl = default_leg(cdo, rates, t, meanloss);
    }
    break;
case T_METHOD_LAURENT_GREGORY_HOMO:
    if(*t_copula != T_COPULA_STUDENT){
        t = init_fine_grid(cdo->dates, p_method[0]);
        cp = init_cond_prob(cdo, cop, t);
        numdef = lg_numdef(cdo, cop, t, cp);
        meanloss = mean_losses_from_numdef(cdo, t, numdef);
        pl = payment_leg(cdo, rates, t, meanloss);
        dl = default_leg(cdo, rates, t, meanloss);
    }
    else {

```

```

    t = init_fine_grid(cdo->dates, p_method[0]);
    cp = init_cond_prob(cdo, cop, t);
    numdef = lg_numdef1(cdo, cop, t, cp);

    meanloss = mean_losses_from_numdef(cdo, t, numdef);
    pl = payment_leg(cdo, rates, t, meanloss);
    dl = default_leg(cdo, rates, t, meanloss);
}
break;
case T_METHOD_HULL_WHITE:
    if(*t_copula != T_COPULA_STUDENT) {
        t = init_fine_grid(cdo->dates, p_method[0]);
        x = init_hom_grid(MINDOUBLE, (1.-recovery[0]), (1.-
recovery[0])/(double) p_method[1]);
        cp = init_cond_prob(cdo, cop, t);
        losses = hw_losses_h(cdo, cop, t, x, cp);
        meanloss = mean_losses(cdo, t, x, losses);
        pl = payment_leg(cdo, rates, t, meanloss);
        dl = default_leg(cdo, rates, t, meanloss);
    }
    else{
        t = init_fine_grid(cdo->dates, p_method[0]);
        x = init_hom_grid(MINDOUBLE, (1.-recovery[0]), (1.-
recovery[0])/(double) p_method[1]);
        cp = init_cond_prob(cdo, cop, t);
        losses = hw_losses_h1(cdo, cop, t, x, cp);
        meanloss = mean_losses(cdo, t, x, losses);
        pl = payment_leg(cdo, rates, t, meanloss);
        dl = default_leg(cdo, rates, t, meanloss);
    }
    break;
case T_METHOD_LAURENT_GREGORY:
    if(*t_copula != T_COPULA_STUDENT){
        t = init_fine_grid(cdo->dates, p_method[0]);
        x = init_hom_grid(MINDOUBLE, (1.-recovery[0]), (1.-
recovery[0])/(double) p_method[1]);
        cp = init_cond_prob(cdo, cop, t);
        losses = lg_losses(cdo, cop, t, x, cp);
        meanloss = mean_losses(cdo, t, x, losses);
        pl = payment_leg(cdo, rates, t, meanloss);
        dl = default_leg(cdo, rates, t, meanloss);
    }

```

```

    }
    else{
        t = init_fine_grid(cdo->dates, p_method[0]);
        x = init_hom_grid(MINDOUBLE, (1.-recovery[0]), (1.-
recovery[0])/(double) p_method[1]);
        cp = init_cond_prob(cdo, cop, t);
        losses = lg_losses1(cdo, cop, t, x, cp);
        meanloss = mean_losses(cdo, t, x, losses);
        pl = payment_leg(cdo, rates, t, meanloss);
        dl = default_leg(cdo, rates, t, meanloss);
    }
    break;
case T_METHOD_MC :
    n_mc = p_method[0];
    pl = mc_payment_leg(cdo, cop, rates, n_mc);
    dl = mc_default_leg(cdo, cop, rates, n_mc);
    break;
case T_METHOD_MC_CV :
    if(*t_copula != T_COPULA_STUDENT) {
        n_mc = p_method[0];
        t = init_fine_grid(cdo->dates, p_method[1]);
        hcdo = homogenize_CDO(cdo);
        cp = init_cond_prob(hcdo, cop, t);
        numdef = lg_numdef(hcdo, cop, t, cp);
        meanloss = mean_losses_from_numdef(hcdo, t, numdef)
;
        hpl = payment_leg(hcdo, rates, t, meanloss);
        hdl = default_leg(hcdo, rates, t, meanloss);
        pl = mc_payment_vc_leg(cdo, cop, rates, n_mc);
        dl = mc_default_vc_leg(cdo, cop, rates, n_mc);
        for (jtr = 0; jtr < cdo->n_tranches-1; jtr++) {
            pl[jtr] = hpl[jtr] + pl[jtr];
            dl[jtr] = hdl[jtr] + dl[jtr];
        }
    }
    else
    {
        n_mc = p_method[0];
        t = init_fine_grid(cdo->dates, p_method[1]);
        hcdo = homogenize_CDO(cdo);
        cp = init_cond_prob(hcdo, cop, t);

```

```

        numdef = lg_numdef1(hcdo, cop, t, cp);
        meanloss = mean_losses_from_numdef(hcdo, t, numd
ef);
        hpl = payment_leg(hcdo, rates, t, meanloss);
        hdl = default_leg(hcdo, rates, t, meanloss);
        pl = mc_payment_vc_leg(cdo, cop, rates, n_mc);
        dl = mc_default_vc_leg(cdo, cop, rates, n_mc);
        for (jtr = 0; jtr < cdo->n_tranches-1; jtr++) {
            pl[jtr] = hpl[jtr] + pl[jtr];
            dl[jtr] = hdl[jtr] + dl[jtr];
        }
    }
    free (hpl); free (hdl);
    free_cdo (&hcdo);
    break;
case T_METHOD_SADDLEPOINT:
    if( (*t_copula != T_COPULA_CLAYTON) && (*t_copula !=
T_COPULA_STUDENT) )
    {
        pl=malloc((cdo->n_tranches-1)*sizeof(double));
        dl=malloc((cdo->n_tranches-1)*sizeof(double));
        t = init_fine_grid(cdo->dates, p_method[0]);
        cp = init_cond_prob(cdo, cop, t);
        tab_sadd=malloc((cdo->n_tranches-1)*sizeof(
double*));
        U=malloc((cdo->n_tranches-1)*sizeof(double**));

        for(jtr=0;jtr<cdo->n_tranches-1;jtr++){
            tab_sadd[jtr]=malloc((t->size)*sizeof(double));
            U[jtr]=malloc((t->size)*sizeof(double*));
        }
        for(jtr=0;jtr<cdo->n_tranches-1;jtr++){
            for(jt=0;jt<t->size;jt++){
                U[jtr][jt]=malloc((cop->size)*sizeof(double**
));
            }
        }

        U=Uoptimal(cdo,cop,t,cp);
        tab_sadd=saddlepoint(cdo,cop,t,cp,U);
    }

```

```

        dl=default_leg_sadd(cdo,rates,t,tab_sadd,cop);

        pl=payment_leg_sadd(cdo,rates,t,tab_sadd,cop);
    }
    else
    {
        printf("NON TREATED CASE{n");
        return(0);
    }
    break;
default:
    printf("Unknown method{n");
    return(0);
}

size_price = n_tranches-1;
for (jtr = 0; jtr < size_price; jtr++) {
    pay_leg[jtr] = (double) pl[jtr];
    def_leg[jtr] = (double) dl[jtr];
    price[jtr] = (def_leg[jtr] / (pay_leg[jtr])) * 10000.;
}

free_cdo(&cdo);

if (numdef != NULL)
{
    for (jt = 0; jt < t->size; jt++) free(numdef[jt]);
    free(numdef);
}
if (losses != NULL)
{
    for (jt = 0; jt < t->size; jt++) free(losses[jt]);
    free(losses);
}
if (cp != NULL) free_cond_prob(cp);
if (meanloss != NULL)
{
    for (jtr = 0; jtr < n_tranches-1; jtr++)

```



```

        free_grid(meanloss[jtr]);
        free(meanloss);
    }

    if (t != NULL) free_grid(t);
    if (x != NULL) free_grid(x);

    free(pl); free(dl);

    free_step_fun(&rates);
    free_copula (&cop);
    free(produit);
    return (0);
}

int premia_interf_price_cdo ( TYPEOPT *ptOpt, TYPEMOD *pt
    Mod, PricingMethod *Met,
                                PnlVect **nominal, PnlVect **
    intensity,
                                int *n_rates, PnlVect **x_ra
    tes, PnlVect **y_rates,
                                int *n_dates, PnlVect **da
    tes, int *n_tranches,
                                int **p_method, int *is_homo)
{
    int i, nvar, n;
    double r, T;

    *n_tranches = ptOpt->tranch.Val.V_PNLVECT->size-1;
    *is_homo = 1;
    /* initialize Results. Have already been allocated in Ini
       t Method. */
    pnl_vect_resize (Met->Res[0].Val.V_PNLVECT, *n_tranches);
    pnl_vect_resize (Met->Res[1].Val.V_PNLVECT, *n_tranches);
    pnl_vect_resize (Met->Res[2].Val.V_PNLVECT, *n_tranches);

    n = ptMod->Ncomp.Val.V_PINT;
    r = ptMod->r.Val.V_DOUBLE;

    T = ptOpt->maturity.Val.V_DATE;

```

```

*n_dates = T * ptOpt->NbPayment.Val.V_INT;
*dates = pnl_vect_create_from_double (*n_dates, 1. / pt
    Opt->NbPayment.Val.V_INT);
pnl_vect_cumsum (*dates);

nvar = 0;
while (Met->Par[nvar].Vtype != PREMIA_NULLTYPE) { nvar++;
    }
*p_method = malloc(sizeof(int)*(nvar+1));
for (i=0; i<nvar; i++) (*p_method)[i] = Met->Par[i].Val.
    V_INT;
(*p_method)[i] = 100; /* subdivision of the losses */

/* interest picewise constant */
*n_rates = 2;
*x_rates = pnl_vect_create_from_double (*n_rates, 0);
*y_rates = pnl_vect_create_from_double (*n_rates, 0);
pnl_vect_set (*x_rates, 1, T);
pnl_vect_set (*y_rates, 1, T*r);

/* nominal */
if (ptOpt->t_nominal.Val.V_ENUM.value == 1)
{
    *nominal = pnl_vect_create_from_double (n, 1./((
double)n);
}
else
{
    VAR *Par;
    *is_homo = 0;
    Par = lookup_premia_enum_par (&(ptOpt->t_nominal), 2)
;
    *nominal = pnl_vect_create_from_file (Par[0].Val.V_
FILENAME);
}

/* intensity */
if (ptMod->t_intensity.Val.V_ENUM.value == 1)
{
    VAR *Par = lookup_premia_enum_par (&(ptMod->t_

```

```
    intensity), 1);
    *intensity = pnl_vect_create_from_double (n, Par[0].
Val.V_PDOUBLE);
}
else
{
    VAR *Par = lookup_premia_enum_par (&(ptMod->t_
intensity), 0);
    *is_homo = 0;
    *intensity = pnl_vect_create_from_file (Par[0].Val.V_
FILENAME);
}

/* check if recovery is constant */
if (ptOpt->t_recovery.Val.V_ENUM.value != 1) *is_homo = 0
;
pnl_rand_init (0, 1, 1);
return OK;
}
#endif //PremiaCurrentVersion
```

References