

Help

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#include "dynamic_stdh.h"
#include "pnl/pnl_matrix.h"
#include "pnl/pnl_integration.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
    (2010+2) //The "#else" part of the code will be freely av
    ailable after the (year of creation of this file + 2)
static int CHK_OPT(Herbertsson)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
int CALC(Herbertsson)(void *Opt, void *Mod, PricingMethod *
    Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
#else

double price_cds(double T,int n,int M,double r,double R, Pn
    lVect*a1){
    /** T maturite du contrat **/
    /** n nbre de subdivisions de l'intervalle [0,T]**/
    /** M nbre de firmes**/
    /** r taux d'interet***/
    /** R recovery en cas de default**/
    /** a vecteur des intensites **/
    PnlMat *Q; /** matrice de transition*/
    PnlVect *l; /**vecteur de perte**/
    PnlVect *a;
    double DL=0; /** default leg selon les tranches**/
    double PL=0; /** payment leg selon les tranches**/
    PnlMat *G, *G1;
    PnlVect *y;
    PnlMat *D;

```

```

PnlMat *F;
int k,i;
double s=0;
double step=T*1./n;
double spread;
int n1,n2,n3,n4,n5;

n1=1+trunc(0.03*M/(1-R));
n2=1+trunc(0.06*M/(1-R));
n3=1+trunc(0.09*M/(1-R));
n4=1+trunc(0.12*M/(1-R));
n5=1+trunc(0.22*M/(1-R));

a=pnl_vect_create_from_double(M+1,0);

pnl_vect_set(a,0,pnl_vect_get(a1,0));

for(i=1;i<n1;i++){
    pnl_vect_set(a,i,pnl_vect_get(a1,1));
}
for(i=n1;i<n2;i++){
    pnl_vect_set(a,i,pnl_vect_get(a1,2));
}
for(i=n2;i<n3;i++){
    pnl_vect_set(a,i,pnl_vect_get(a1,3));
}
for(i=n3;i<n4;i++){
    pnl_vect_set(a,i,pnl_vect_get(a1,4));
}
for(i=n4;i<n5;i++){
    pnl_vect_set(a,i,pnl_vect_get(a1,5));
}
for(i=n5;i<M+1;i++){
    pnl_vect_set(a,i,pnl_vect_get(a1,6));
}

/**Definition de la matrice de transition**/
Q=pnl_mat_create_from_double(M+1,M+1,0);
D=pnl_mat_create_from_double(M+1,M+1,0);
F=pnl_mat_create_from_double(M+1,M+1,0);
G=pnl_mat_create_from_double(M+1,M+1,0);

```

```

G1=pnl_mat_create_from_double(M+1,M+1,0);

for(k=0;k<M;k++){
    s=s+pnl_vect_get(a,k);
    pnl_mat_set(Q,k,k,-(M-k)*s);
    pnl_mat_set(Q,k,k+1,(M-k)*s);
}
/** Iteration pour le calcul du prix de chaque tranche CDO**/
l=pnl_vect_create_from_double(M+1,0);
y=pnl_vect_create_from_double(M+1,0);
pnl_mat_clone(D,Q);

for(i=0;i<M+1;i++){
    pnl_mat_set(D,i,i,pnl_mat_get(D,i,i)-r);
}
/**calcul de l'inverse de Q-rI**/
pnl_mat_upper_inverse(G,D);

/**Definition du vecteur l pour la p-ieme tranche CDO**/
for(i=0;i<M+1;i++){
    pnl_vect_set(l,i,i*(1-R)*1./M);
}

for(k=1;k<n+1;k++){
    /**Calcul du DL**/
    pnl_mat_clone(D,Q);
    pnl_mat_mult_double(D,k*step);
    pnl_mat_exp(F,D);

    pnl_mat_mult_vect_inplace(y,F,l);
    PL+=step*exp(-r*step*k)*(1-pnl_vect_get(y,0)*1./(1-R));

}
pnl_mat_mult_double(F,exp(-r*T));
pnl_mat_clone(D,F);

for(i=0;i<M+1;i++){
    pnl_mat_set(F,i,i,pnl_mat_get(F,i,i)-1);
}
pnl_mat_mult_mat_inplace(G1,F,G);

```

```

    pnl_mat_clone(G,G1);
    pnl_mat_mult_double(G,r);
    pnl_mat_plus_mat(G,D);
    pnl_mat_mult_vect_inplace(y,G,1);
    DL=pnl_vect_get(y,0);

    /** Calcul du spread exprime en bps**/
    spread=10000*DL/PL;

    pnl_mat_free (&G);
    pnl_mat_free (&G1);
    pnl_mat_free (&D);
    pnl_mat_free (&Q);
    pnl_mat_free (&F);
    pnl_vect_free (&y);
    pnl_vect_free (&l);
    pnl_vect_free (&a);
    return(spread);
}

int CALC(Herbertsson)(void *Opt, void *Mod, PricingMethod *
    Met)
{
    TYPEOPT      *ptOpt      = (TYPEOPT*)Opt;
    TYPEMOD      *ptMod      = (TYPEMOD*)Mod;
    int          n, sub;
    double        recovery, r, maturity;
    PnlVect *a;    /** vecteur des intensites */

    n = ptMod->Ncomp.Val.V_PINT;
    r = ptMod->r.Val.V_DOUBLE;
    maturity = ptOpt->Maturity.Val.V_DATE;
    recovery = ptOpt->Recovery.Val.V_DOUBLE;
    sub = (int) (ptOpt->NbPayement.Val.V_PINT * maturity);

    a=pnl_vect_create(7);
    /** Intensites associees***/
    pnl_vect_set(a,0,0.0033);
    pnl_vect_set(a,1,0.00164);
    pnl_vect_set(a,2,0.00845);
    pnl_vect_set(a,3,0.0145);

```

```
pnl_vect_set(a,4,0.00864);
pnl_vect_set(a,5,0.0124);
pnl_vect_set(a,6,0.0514);

Met->Res[0].Val.V_DOUBLE = price_cds(maturity,sub,n,r,
    recovery,a);
pnl_vect_free (&a);
return OK;
}

static int CHK_OPT(Herbertsson)(void *Opt, void *Mod)
{
    return OK;
}

#endif //PremiaCurrentVersion
static int MET(Init)(PricingMethod *Met,Option *Opt)
{
    if ( Met->init == 0)
    {
        Met->init=1;
        Met->HelpFilenameHint = "herbertsson_cds";
    }
    return OK;
}

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

References