

## Help

```
#include "hullwhite2d_std.h"

int MOD_OPT(ChkMix)(Option *Opt,Model *Mod)
{
    TYPEOPT* ptOpt=( TYPEOPT*)(Opt->TypeOpt);
    TYPEMOD* ptMod=( TYPEMOD*)(Mod->TypeModel);
    int status=OK;
    if ((strcmp(Opt->Name,"ZeroCouponCallBondEuro")==0) || (
        strcmp(Opt->Name,"ZeroCouponPutBondEuro")==0) || (strcmp(Opt-
        >Name,"ZeroCouponCallBondAmer")==0) || (strcmp(Opt->Name,"
        ZeroCouponPutBondAmer")==0))
    {
        if ((ptOpt->OMaturity.Val.V_DATE)<=(ptMod->T.Val.V_DA
        TE))
        {
            Fprintf(TOSCREENANDFILE,"Current date greater than
            maturity!\n");
            status+=1;
        }
        if((ptOpt->BMaturity.Val.V_DATE)<=(ptOpt->OMaturity.
        Val.V_DATE))
        {
            Fprintf(TOSCREENANDFILE,"Option maturity greater than
            Bond maturity!\n");
            status+=1;
        }
    }
    if ((strcmp(Opt->Name,"ZeroCouponBond")==0))
    {
        if ((ptOpt->BMaturity.Val.V_DATE)<=(ptMod->T.Val.V_DA
        TE))
        {
            Fprintf(TOSCREENANDFILE,"Current date greater than
            maturity!\n");
            status+=1;
        }
    }
    if ((strcmp(Opt->Name,"PayerSwaption")==0)|| (strcmp(Opt-
    >Name,"ReceiverSwaption")==0)|| (strcmp(Opt->Name,"
    PayerBermudanSwaption")==0)|| (strcmp(Opt->Name,"
```

```

ReceiverBermudanSwaption")==0))
    if((ptOpt->BMaturity.Val.V_DATE)<=(ptOpt->OMaturity.
Val.V_DATE))
{
    Fprintf(TOSCREENANDFILE,"Option maturity greater than
Bond maturity!\n");
    status+=1;
}

if ((strcmp(Opt->Name,"Floor")==0)|| (strcmp(Opt->Name,"    Cap")==0))
{
    if ((ptOpt->FirstResetDate.Val.V_DATE)<=(ptMod->T.Val
.V_DATE))
{
    Fprintf(TOSCREENANDFILE,"Current date greater than fir
st coupon date!\n");
    status+=1;
}

    if ((ptOpt->FirstResetDate.Val.V_DATE)>=(ptOpt->BMatu
rity.Val.V_DATE))
{
    Fprintf(TOSCREENANDFILE,"First reset date greater than
contract maturity!\n");
    status+=1;
}
}

return status;
}

extern PricingMethod MET(TR_ZBOHW2D);
extern PricingMethod MET(CF_ZCBONDHW2D);
extern PricingMethod MET(CF_ZBCALLHW2D);
extern PricingMethod MET(CF_ZBPUTHW2D);
extern PricingMethod MET(CF_CAPHW2D);
extern PricingMethod MET(CF_FLOORHW2D);
extern PricingMethod MET(CF_EuropeanSwaption_HW2D);
extern PricingMethod MET(TR_SWAPTIONHW2D);
extern PricingMethod MET(TR_BERMUDIANSWAPTIONHW2D);
extern PricingMethod MET(TR_CAPFLOORHW2D);
extern PricingMethod MET(TR_ZCBONDHW2D);

```

```

PricingMethod* MOD_OPT(methods)[]={
    &MET(TR_ZBOHW2D),
    &MET(CF_ZCBONDHW2D),
    &MET(CF_ZBCALLHW2D),
    &MET(CF_ZBPUTHW2D),
    &MET(CF_CAPHW2D),
    &MET(CF_FLOORHW2D),
    &MET(CF_EuropeanSwaption_HW2D),
    &MET(TR_SWAPTIONHW2D),
    &MET(TR_BERMUDIANSWAPTIONHW2D),
    &MET(TR_CAPFLOORHW2D),
    &MET(TR_ZCBONDHW2D),
    NULL
};

///  

DynamicTest* MOD_OPT(tests)[]={
    NULL
};

Pricing MOD_OPT(pricing)={
    ID_MOD_OPT,
    MOD_OPT(methods),
    MOD_OPT(tests),
    MOD_OPT(ChkMix)
};

```

## References