

[Help](#)

```
#include "lmm_jump1d_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,"ZCBond")==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(AP_GM);
extern PricingMethod MET(MC_GM);

PricingMethod* MOD_OPT(methods)[]={
    &MET(AP_GM),
    &MET(MC_GM),
    NULL
};

```

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

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

## References