Help

```
#include "pnl/pnl_integration.h"

#include  "
href../../../mod/lmm_stochvol_piterbarg/lmm_stochvol_piterbarg_stdi/lmm_stochvol
#include "
href../../../common/math/lmm_stochvol_piterbarg/lmm_stochvol_piterbarg_h_src.pdf
#include "
href../../../common/math/lmm_stochvol_piterbarg/ap_averagingtech_lmmpit_h_src.pd
#include "
href../../../common/enums_h_src.pdfenums.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2010+2) //The "#els
static int CHK_OPT(AP_Swaption_LmmPit)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(AP_Swaption_LmmPit)(void *Opt, void *Mod, PricingMethod *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else

int func_premia_swaption(int InitYieldCurve_flag, double R_flat, char *curve, do
                         double SkewsParams_a, double SkewsParams_b, double Skew
                         double VolsParams_a, double VolsParams_b, double VolsPa
                         double Tn, double Tm, double period, double swaption_st
{
  int NbrVolFactors = 1;
  StructLmmPiterbarg *LmmPiterbarg;
  PnlMat *SkewsParams = pnl_mat_create_from_list(4, 1, SkewsParams_a, SkewsParam
  PnlMat *VolsParams = pnl_mat_create_from_list(4, 1, VolsParams_a, VolsParams_b

  LmmPiterbarg = SetLmmPiterbarg(InitYieldCurve_flag, R_flat, curve, period, Tm,


  *price = cf_lmm_stochvol_piterbarg_swpt(LmmPiterbarg, Tn, Tm, period, swaption

  FreeLmmPiterbarg(&LmmPiterbarg);
```

```
  pnl_mat_free(&SkewsParams);
  pnl_mat_free(&VolsParams);

  return OK;
}

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

  int Payer_Receiver = 1;
  if (((ptOpt->PayOff.Val.V_NUMFUNC_1)->Compute) == &Call)
    {
      Payer_Receiver = -1;
    }

  return func_premia_swaption(ptMod->Flag_InitialYieldCurve.Val.V_INT,
                              MOD(GetYield)(ptMod),
                              MOD(GetCurve)(ptMod),
                              ptMod->Var_SpeedMeanReversion.Val.V_PDOUBLE,
                              ptMod->Var_Volatility.Val.V_PDOUBLE,
                              ptMod->SkewsParams_a.Val.V_PDOUBLE,
                              ptMod->SkewsParams_b.Val.V_PDOUBLE,
                              ptMod->SkewsParams_c.Val.V_PDOUBLE,
                              ptMod->SkewsParams_d.Val.V_PDOUBLE,
                              ptMod->VolsParams_a.Val.V_PDOUBLE,
                              ptMod->VolsParams_b.Val.V_PDOUBLE,
                              ptMod->VolsParams_c.Val.V_PDOUBLE,
                              ptMod->VolsParams_d.Val.V_PDOUBLE,
                              ptOpt->OMaturity.Val.V_DATE - ptMod->T.Val.V_DATE,
                              ptOpt->BMaturity.Val.V_DATE - ptMod->T.Val.V_DATE,
                              ptOpt->ResetPeriod.Val.V_DATE,
                              ptOpt->FixedRate.Val.V_PDOUBLE,
                              ptOpt->Nominal.Val.V_PDOUBLE,
                              Payer_Receiver,
                              Met->Par[0].Val.V_ENUM.value,
                              &(Met->Res[0].Val.V_DOUBLE));

}
```

```
static int CHK_OPT(AP_Swaption_LmmPit)(void *Opt, void *Mod)
{
  if ((strcmp(((Option *)Opt)->Name, "PayerSwaption") == 0) || (strcmp(((Option
    return OK;
  else
    return WRONG;
}
#endif //PremiaCurrentVersion


static int MET(Init)(PricingMethod *Met, Option *Opt)
{
  if (Met->init == 0)
    {
      Met->init = 1;
      Met->Par[0].Val.V_ENUM.value = 0;
      Met->Par[0].Val.V_ENUM.members = &PremiaEnumAveraging;
    }

  return OK;
}


PricingMethod MET(AP_Swaption_LmmPit) =
{
  "AP_Swaption_LmmPit",
  {
    {"Averaging Vol", ENUM, {100}, ALLOW},
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CALC(AP_Swaption_LmmPit),
  {{"Price", DOUBLE, {100}, FORBID}, {" ", PREMIA_NULLTYPE, {0}, FORBID}},
  CHK_OPT(AP_Swaption_LmmPit),
  CHK_ok,
  MET(Init)
} ;
```