Help

```
#include <cstdlib>
#include <string>
#include "math/ImportanceSampling_jl/src/wrapper.hpp"

using namespace std;

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2015+2) //The "#els
static int CHK_OPT(MC_FixedAsian_IS)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(MC_FixedAsian_IS)(void *Opt, void *Mod, PricingMethod *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else

extern "C" {
#include "kou1d_pad.h"
#include "enums.h"

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

    double r, divid, time_spent, pseudo_spot, pseudo_strike;
    double t_0, T_0;

    r = log(1. + ptMod->R.Val.V_DOUBLE / 100.);
    divid = log(1. + ptMod->Divid.Val.V_DOUBLE / 100.);

    T_0 = ptMod->T.Val.V_DATE;
    t_0 = (ptOpt->PathDep.Val.V_NUMFUNC_2)->Par[0].Val.V_PDOUBLE;


    if (T_0 < t_0)
      {
        Fprintf(TOSCREEN, "T_0 < t_0, untreated case\ n\ n\ n");
```

```
      return WRONG;
    }
  /* Case t_0 <= T_0 */
  else
    {
      time_spent = (ptMod->T.Val.V_DATE - (ptOpt->PathDep.Val.V_NUMFUNC_2)->Pa
      pseudo_spot = (1. - time_spent) * ptMod->S0.Val.V_PDOUBLE;
      pseudo_strike = (ptOpt->PayOff.Val.V_NUMFUNC_2)->Par[0].Val.V_PDOUBLE -

      Param P;
      bool poisson = false, poisson_only = false;
      PnlRng *rng = pnl_rng_create(Met->Par[3].Val.V_ENUM.value);
      pnl_rng_sseed(rng, 0);
      std::vector<double> lambda_m(1, ptMod->LambdaMinus.Val.V_PDOUBLE);
      std::vector<double> lambda_p(1, ptMod->LambdaPlus.Val.V_PDOUBLE);
      std::vector<double> jump_intensity(1, ptMod->Lambda.Val.V_PDOUBLE);
      std::vector<double> pos_proba_jump(1, ptMod->P.Val.V_PDOUBLE);
      std::vector<double> spot(1, pseudo_spot);
      std::vector<double> volatility(1, ptMod->Sigma.Val.V_PDOUBLE);
      std::vector<double> dividend_rate(1, divid);


      if ((ptOpt->PayOff.Val.V_NUMFUNC_2)->Compute == &Call_OverSpot2)
        P.insert("option type", T_STRING, string("asiancall"));
      else
        P.insert("option type", T_STRING, string("asianput"));

      P.insert("model type", T_STRING, string("kou"));
      P.insert("option size", T_INT, 1);
      P.insert("poisson size", T_INT, 1);
      P.insert("jump intensity", T_VECTOR, jump_intensity);
      P.insert("lambda_p", T_VECTOR, lambda_p);
      P.insert("lambda_m", T_VECTOR, lambda_m);
      P.insert("positive jump probability", T_VECTOR, pos_proba_jump);
      P.insert("strike", T_DOUBLE, pseudo_strike);
      P.insert("spot", T_VECTOR, spot);
      P.insert("maturity", T_DOUBLE, ptOpt->Maturity.Val.V_DATE - ptMod->T.Val
      P.insert("volatility", T_VECTOR, volatility);
      P.insert("interest rate", T_DOUBLE, r);
      P.insert("dividend rate", T_VECTOR, dividend_rate);
      P.insert("correlation", T_DOUBLE, 0.);
```

```
        P.insert("step size", T_DOUBLE, 0.5);
        P.insert("timestep number", T_INT, Met->Par[0].Val.V_PINT);
        P.insert("sample number", T_INT, Met->Par[1].Val.V_PINT);

        if (Met->Par[2].Val.V_ENUM.value == 1)
          {
            poisson_only = false;
            poisson = false;
          }
        else if (Met->Par[2].Val.V_ENUM.value == 2)
          {
            poisson_only = true;
            poisson = true;
          }
        else if (Met->Par[2].Val.V_ENUM.value == 3)
          {
            poisson_only = false;
            poisson = true;
          }
        MonteCarloWrapper(rng, P, false, false, true, false, false, poisson_only

        // Do not delete the vectors inserted in the Param P because they are ha
        pnl_rng_free(&rng);
        return OK;
      }
  }


  static int CHK_OPT(MC_FixedAsian_IS)(void *Opt, void *Mod)
  {
    if ((strcmp(((Option *)Opt)->Name, "AsianCallFixedEuro") == 0) || (strcmp(((
      return OK;
    return WRONG;
  }

#endif //PremiaCurrentVersion
static int MET(Init)(PricingMethod *Met, Option *Opt)
{
  if (Met->init == 0)
    {
  Met->HelpFilenameHint = "MC_FixedAsian_IS";
```

```
      Met->init = 1;
      Met->Par[0].Val.V_PINT = 52;
      Met->Par[1].Val.V_PINT = 30000;
      Met->Par[2].Val.V_ENUM.members = &PremiaEnumISType;
      Met->Par[2].Val.V_ENUM.value = 1;
      Met->Par[3].Val.V_ENUM.members = &PremiaEnumMCRNGs;
      Met->Par[3].Val.V_ENUM.value = 0;
    }
  return OK;
}

PricingMethod MET(MC_FixedAsian_IS) =
{
  "MC_FixedAsian_IS_Lelong",
  {
    {"Nb.of Monitoring Dates", PINT, {2000}, ALLOW, SETABLE},
    {"Nb.of Samples", PINT, {20000}, ALLOW, SETABLE},
    {"Type of IS", ENUM, {0}, ALLOW, SETABLE},
    {"Random Generator", ENUM, {0}, ALLOW, SETABLE},
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CALC(MC_FixedAsian_IS),
  {
    {"Price", DOUBLE, {100}, FORBID},
    {"Std_dev", DOUBLE, {100}, FORBID},
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CHK_OPT(MC_FixedAsian_IS),
  CHK_ok,
  MET(Init)
};

}
```