Help

```
#include "bs1d_default_stdr.h"
#include "pnl/pnl_cdf.h"
#include "pnl/pnl_mathtools.h"
#include "pnl/pnl_vector.h"
#include "pnl/pnl_matrix.h"


#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2016+2) //The "#els

static int CHK_OPT(AP_CVABenTaaritLapeyre)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(AP_CVABenTaaritLapeyre)(void *Opt, void *Mod, PricingMethod *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else


static int ForwardEPE(int N, double S, double K, double T, double r, double q, d
{
    int i;
    double Ti, y_star;
    double delta_star = exp(-r*T)*K;

    for (i = 0; i < EPE_ODE->size; i++)
    {
        if (i == 0)
            LET(EPE_ODE, i) = 0.;
        else
        {
            Ti = (T*(1 - 1/(N-1)) / (N-1))*i;
LET(vect_time, i)=Ti;
            y_star = log(K) - (r-q)*(T-Ti);
            LET(EPE_ODE, i) = exp(-SQR(y_star - (log(S) + (r - q - 0.5*SQR(sigma
                                ( 2.*SQR(sigma*sqrt(Ti)) )) /
                            ( sigma*sqrt(Ti)*sqrt(M_2PI) );
            LET(EPE_ODE, i) = SQR(sigma)*delta_star*GET(EPE_ODE, i);
```

```
        }
    }

    pnl_vect_cumsum(EPE_ODE);
    pnl_vect_mult_scalar(EPE_ODE, 0.5*T/(N-1));
    pnl_vect_plus_scalar(EPE_ODE, MAX( exp(-q*T)*(S - exp(-(r-q)*T)*K) , 0 ));

    return OK;
}

static int  EPE_EFRA(double lambda, double recovery,double S, double K, double T
{
    int i;
double sum=0;
    PnlVect* EPE_ODE = pnl_vect_create_from_zero(N);
PnlVect* vect_time = pnl_vect_create_from_zero(N);

    ForwardEPE(N, S, K, T, r, q, sigma, EPE_ODE,vect_time);
    for (i = 1; i < N; i++)
    {
  sum+=(1-recovery)*(exp(-lambda*GET(vect_time, i-1))-exp(-lambda*GET(vect_time,
    }

*ptprice=sum;

    pnl_vect_free(&EPE_ODE);
    pnl_vect_free(&vect_time);

return OK;
}

static int CALC(AP_CVABenTaaritLapeyre)(void *Opt, void *Mod, PricingMethod *Met
{
  TYPEOPT *ptOpt = (TYPEOPT *)Opt;
  TYPEMOD *ptMod = (TYPEMOD *)Mod;
  double r, divid;

  r = ptMod->Interest.Val.V_DOUBLE;
  divid=0;

  return  EPE_EFRA(ptMod->Intensity.Val.V_PDOUBLE,ptMod->Recovery.Val.V_DOUBLE,p
```

```
}

static int CHK_OPT(AP_CVABenTaaritLapeyre)(void *Opt, void *Mod)
{
  if (strcmp(((Option *)Opt)->Name, "CVA_CallEuro") != 0) return FAIL;
  return OK;
}

#endif

static int MET(Init)(PricingMethod *Met, Option *Opt)
{
  if (Met->init == 0)
    {
      Met->init = 1;
      Met->HelpFilenameHint = "ap_CVABenTaaritLapeyre";
      Met->Par[0].Val.V_PINT = 10000;
    }
  return OK;
}


PricingMethod MET(AP_CVABenTaaritLapeyre) =
{
  "AP_CVABenTaaritLapeyre",
  {
    {"Numbers of Integration Steps", PINT, {22}, ALLOW},
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CALC(AP_CVABenTaaritLapeyre),
  {
    {"CVA_CallEuro", DOUBLE, {0}, FORBID},
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CHK_OPT(AP_CVABenTaaritLapeyre),
  CHK_ok,
  MET(Init)
};
```