Help

```
#include  "merhes1d_std.h"

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

static double d1(double x, double t, double s, double K, double r, double T)
{
  double d = (log(x / K) + (r + s * s / 2) * (T - t)) / (s * sqrt(T - t));
  return d;
}

static double H(double t, double x, double v, double K, double r, double T)
{
  double a, d, HH;
  a = d1(x, t, v, K, r, T) * d1(x, t, v, K, r, T);
  d = v * sqrt((T - t) * 2 * M_PI);
  HH = exp(-a / 2) / d * x * (1 - d1(x, t, v, K, r, T) / v / sqrt(T));
  return HH;
}

static double diffH(double v, double T, double S, double K, double r)
{
  return (-0.5 / pow(v, 3) * pow(2, 0.5) / pow(M_PI, 0.5) / pow(T, 0.5) * (log(S
}


int ApAlosBates(double S, NumFunc_1  *p, double T, double r, double divid, doubl
{
  int flag_call;
```

```
double K, prix, delta, price_bs, delta_bs, price_j, delta_j;
double v0et, I, d, d2BS;

K = p->Par[0].Val.V_PDOUBLE;

if ((p->Compute) == &Call)
  flag_call = 1;
else
  flag_call = 0;;

//Calculation of the quantity denote by v0* in the paper
v0et = sqrt(theta + 1 / (kappa * T) * (v0 - theta) * (1 - exp(-kappa * T)));

// Calculation of the quantity denote by I in the paper
d = diffH(v0et, T, S, K, r);
I = sigma / kappa / kappa * (theta * (kappa * T - 2) + v0 + exp(-kappa * T) *
//calculation of dšBS/dxš
d2BS = 1 / sqrt(T * 2 * M_PI) / v0et * exp(-d1(S, 0, v0et, K, r, T) * d1(S, 0,
pnl_cf_call_bs(S + m + pow(v, 2.)*T / 2, K, T, r, divid, sqrt(pow(v0et, 2) + v

if (flag_call == 1)
  {
    pnl_cf_call_bs(S, K, T, r, divid, v0et, &price_bs, &delta_bs);
    prix = price_bs + rho / 2.*H(0, S, v0et, K, r, T) * I;
    prix = prix - (exp(m + pow(v, 2) / 2) - 1) * lambda * T * delta_bs - lambd
    delta = delta_bs + rho / 2 * I * d;
    delta = delta + lambda * T * delta_j - lambda * T * delta_bs - (exp(m + po
  }
else
  {
    pnl_cf_put_bs(S, K, T, r, divid, v0et, &price_bs, &delta_bs);
    prix = price_bs + rho / 2 * H(0, S, v0et, K, r, T) * I;
    prix = prix + (exp(m + pow(v, 2) / 2) - 1) * lambda * T * delta_bs - lambd
    delta = delta_bs + rho / 2 * I * d;
    delta = delta - lambda * T * delta_j + lambda * T * delta_bs
            - (exp(m + pow(v, 2) / 2) - 1) * lambda * T * d2BS;
  }

/* Price*/
*ptprice = prix;
```

```
  /* Delta */
  *ptdelta = delta;

  return OK;
}

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

  if (ptMod->Sigma.Val.V_PDOUBLE == 0.0)
    {
      Fprintf(TOSCREEN, "BLACK-SHOLES MODEL\ n\ n\ n");
      return WRONG;
    }
  else
    {
      r = log(1. + ptMod->R.Val.V_DOUBLE / 100.);
      divid = log(1. + ptMod->Divid.Val.V_DOUBLE / 100.);

      return ApAlosBates(ptMod->S0.Val.V_PDOUBLE,
                         ptOpt->PayOff.Val.V_NUMFUNC_1,
                         ptOpt->Maturity.Val.V_DATE - ptMod->T.Val.V_DATE,
                         r,
                         divid, ptMod->Sigma0.Val.V_PDOUBLE
                         , ptMod->MeanReversion.Val.V_PDOUBLE,
                         ptMod->LongRunVariance.Val.V_PDOUBLE,
                         ptMod->Sigma.Val.V_PDOUBLE,
                         ptMod->Rho.Val.V_PDOUBLE,
                         ptMod->Mean.Val.V_PDOUBLE,
                         ptMod->Variance.Val.V_PDOUBLE,
                         ptMod->Lambda.Val.V_PDOUBLE,
                         &(Met->Res[0].Val.V_DOUBLE),
                         &(Met->Res[1].Val.V_DOUBLE)
                        );
    }

}
```

```
static int CHK_OPT(AP_Alos_Bates)(void *Opt, void *Mod)
{
  if ((strcmp(((Option *)Opt)->Name, "CallEuro") == 0)
      || (strcmp(((Option *)Opt)->Name, "PutEuro") == 0))

    return OK;
  return WRONG;
}

#endif //PremiaCurrentVersion
static int MET(Init)(PricingMethod *Met, Option *Opt)
{
  if (Met->init == 0)
    {
      Met->init = 1;
    }

  return OK;
}

PricingMethod MET(AP_Alos_Bates) =
{
  "AP_Alos_Bates",
  {{" ", PREMIA_NULLTYPE, {0}, FORBID}},
  CALC(AP_Alos_Bates),
  { {"Price", DOUBLE, {100}, FORBID},
    {"Delta", DOUBLE, {100}, FORBID} ,
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CHK_OPT(AP_Alos_Bates),
  CHK_ok,
  MET(Init)
};
```