Help

```
#include  "
href../../../mod/bs1d/bs1d_pad/bs1d_pad_h_src.pdfbs1d_pad.h"

int Levy_FixedAsian(double pseudo_stock, double pseudo_strike, NumFunc_2  *po, d
{

  double m1, m2, m, v, d1, d2, esp, nd1, nd2;
  double CTtK, PTtK, Dlt, Plt;
  double new_r, new_sigma;

  /*Computation of the first two moments*/
  new_r = (r - divid) * t;
  new_sigma = sigma * sqrt(t);
  m1 = Moments(1, new_r, new_sigma, 1);
  m2 = Moments(2, new_r, new_sigma, 1);

  /*Fit the parameters m,v of lognormal distribution*/
  m = 2.0 * log(m1) - log(m2) / 2.0;
  v = sqrt(log(m2) - 2.0 * log(m1));

  /*Adjusted input for Black-Scholes Formula*/
  d1 = (log(pseudo_stock / pseudo_strike) + m + SQR(v)) / v;
  d2 = d1 - v;
  esp = m + SQR(v) / 2.0 - (r - divid) * t;
  nd1 = cdf_nor(d1);
  nd2 = cdf_nor(d2);

  /*  Call Price */
  CTtK = pseudo_stock * exp(-divid * t) * exp(esp) * nd1 - exp(-r * t) * pseudo_

  /*  Put Price from Parity*/
  if (r == divid)
    PTtK = CTtK + pseudo_strike * exp(-r * t) - pseudo_stock * exp(-r * t);
  else
    PTtK = CTtK + pseudo_strike * exp(-r * t) - pseudo_stock * exp(-r * t) * (ex

  /*Delta for call option*/
  Dlt = exp(esp) * nd1 * exp(-divid * t);
```

```
  /*Delta for put option*/
  if (r == divid)
    Plt = Dlt - exp(-r * t);
  else
    Plt = Dlt - exp(-r * t) * (exp((r - divid) * t) - 1.0) / (t * (r - divid));

  /*Price*/
  if ((po->Compute) == &Call_OverSpot2)
    *ptprice = CTtK;
  else
    *ptprice = PTtK;

  /*Delta */
  if ((po->Compute) == &Call_OverSpot2)
    *ptdelta = Dlt;
  else
    *ptdelta = Plt;

  return OK;
}

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

  int return_value;
  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_value = WRONG;
```

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

      if (pseudo_strike <= 0.)
        {
          Fprintf(TOSCREEN, "ANALYTIC FORMULA\ n\ n\ n");
          return_value = Analytic_KemnaVorst(pseudo_spot, pseudo_strike, time_sp
        }
      else
        return_value = Levy_FixedAsian(pseudo_spot, pseudo_strike, ptOpt->PayOff
    }
  return return_value;

}

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

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

  return OK;
}

PricingMethod MET(AP_FixedAsian_Levy) =
{
  "AP_FixedAsian_Levy",
  {{" ", PREMIA_NULLTYPE, {0}, FORBID}},
```

```
  CALC(AP_FixedAsian_Levy),
  {{"Price", DOUBLE, {100}, FORBID}, {"Delta", DOUBLE, {100}, FORBID} , {" ", PR
  CHK_OPT(AP_FixedAsian_Levy),
  CHK_ok,
  MET(Init)
};
```