Help

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

#define NPOINTS 60
#define JMAX 40

/*This is the integrand in the formula 3.4 of Thompson*/
static double GetGamma(double trialGamma, double r, double sg, double t)
{

  double a = (r - sg * sg / 2.0);

  return (exp(a * t + 0.5 * sg * sg * t * (1.0 - 3.0 * t * (1.0 - t / 2.0) * (1.
              3.0 * sg * trialGamma * t * (1.0 - t / 2.0)));
}

/*This is the integral of the formula 3.4 in Thompson*/
static double integragamma(double trialGamma, double spot, double strike, double
{
  int i;
  double sum, t[NPOINTS + 1], w[NPOINTS + 1];

  sum = 0.0;
  gauleg(0.0, 1.0, t, w, NPOINTS);
  for (i = 1; i <= NPOINTS; i++)
    sum += w[i] * GetGamma(trialGamma, r, sg, t[i]);

  return spot * sum - strike;
}

/*We obtain the optimal value of gamma using bisection method*/
static double findgamma(double spot, double strike, double r, double sg, double
{
  int j;
  double dg, f, fmid, gmid, rtb;

  f = integragamma(gmin, spot, strike, r, sg);
  fmid = integragamma(gmax, spot, strike, r, sg);
```

```
  rtb = f < 0.0 ? (dg = gmax - gmin, gmin) : (dg = gmin - gmax, gmax);
  for (j = 1; j <= JMAX; j++)
    {
      fmid = integragamma((gmid = rtb + (dg *= 0.5)), spot, strike, r, sg);
      if (fmid <= 0.0) rtb = gmid;
      if (fabs(dg) < gacc || fmid == 0.0) return rtb;
    }

  return 0.0;
}


/*This is the function to be integrated on order to get lower bound in Thompson*
static double intlowerbound(double t, double spot, double strike, double r, doub
{
  double a = (r - sg * sg / 2.0);
  double gfind = findgamma(spot, strike, r, sg, gmin, gmax, gacc);
  double arg1 = (-gfind + sg * t * (1 - t / 2.0)) / (1.0 / sqrt(3.));

  return spot * exp(a * t + sg * sg * t / 2.0) * cdf_nor(arg1);
}


static int ThompsonLow_FixedAsian(double pseudo_stock, double pseudo_strike, Num
{
  int i;
  double sum, sum_delta, inc, gmin, gmax, gacc, gfind, arg2, tw[NPOINTS + 1], w[
  double CTtK, PTtK, Dlt, Plt;

  /*Increment for the Delta*/
  inc = 1.0e-3;

  /*Scaling of the parameters*/
  new_r = (r - divid) * t;
  new_sigma = sigma * sqrt(t);

  /*Integrate, using the Laguerre quadrature, for obtaining the lower bound  */
  gauleg(0.0, 1.0, tw, w, NPOINTS);
  gmin = -10.;
  gmax = 10.;
  gacc = 1.0e-8;
```

```
sum = 0.0;
sum_delta = 0.;
for (i = 1; i <= NPOINTS; i++)
  {
    sum += w[i] * intlowerbound(tw[i], pseudo_stock, pseudo_strike, new_r, new
    sum_delta += w[i] * intlowerbound(tw[i], pseudo_stock * (1. + inc), pseudo
  }

gfind = findgamma(pseudo_stock, pseudo_strike, new_r, new_sigma, gmin, gmax, g
arg2 = -gfind / (1.0 / sqrt(3.0));

/*  Call Price */
CTtK = exp(-r * t) * (sum - pseudo_strike * cdf_nor(arg2));

/*  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*/
gfind = findgamma(pseudo_stock * (1. + inc), pseudo_strike, new_r, new_sigma,
arg2 = -gfind / (1.0 / sqrt(3.0));
Dlt = (exp(-r * t) * (sum_delta - pseudo_strike * cdf_nor(arg2)) - CTtK) / (ps

/*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_ThompsonLow)(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 = ThompsonLow_FixedAsian(pseudo_spot, pseudo_strike, ptOpt-
    }
```

```
  return return_value;
}

static int CHK_OPT(AP_FixedAsian_ThompsonLow)(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_ThompsonLow) =
{
  "AP_FixedAsian_ThompsonLow",
  {{" ", PREMIA_NULLTYPE, {0}, FORBID}},
  CALC(AP_FixedAsian_ThompsonLow),
  {{"Price", DOUBLE, {100}, FORBID}, {"Delta", DOUBLE, {100}, FORBID} , {" ", PR
  CHK_OPT(AP_FixedAsian_ThompsonLow),
  CHK_ok,
  MET(Init)
};
#undef NPOINTS
#undef JMAX
```