```
   Help
#include  "bs1d_pad.h"

#define NPOINTS 60

/*Formula 4.4 of Thompson */
static double Sqrtvarnt(double spot, double strike, double
    r, double sigma, double t)
{
  double alpha = r-sigma*sigma/2.0;
  double eat   = exp(alpha*t);
  double ct    = spot*eat*sigma - strike*sigma;
  double t2    = strike*sigma;

  return sqrt( ct*ct*t + 2.0*t2*ct* t*(1.0-0.5*t)  + t2*t2/
    3.0 );
}

/*This is the integral of the formula 4.4 in Thompson */
static double IntegraSqrtvarnt(double spot, double strike,
    double r, double sigma)
{
  int i;
  double sum, t[NPOINTS+1],w[NPOINTS+1];

  gauleg(0.0, 1.0, t, w, NPOINTS);
  sum=0.0;
  for (i=1;i<=NPOINTS;i++)
    sum += w[i]*Sqrtvarnt(spot, strike, r, sigma, t[i]);

  return sum;
}

/* Integrand for upper bound */
double f_upper(double spot, double strike, double r,
    double sigma, double t, double x)
{
  double alpha = r-sigma*sigma/2.0;
  double eat   = exp(alpha*t);
  double ct = spot*eat*sigma - strike*sigma;
  double t2 = strike*sigma;
```

```
  double sqrtvarnt = sqrt( ct*ct*t +2.0*ct*t2* t*(1.0-0.5*
    t) + t2*t2/3.0);
  double gamma = (strike - spot*(exp(alpha) - 1.0)/alpha)/
    IntegraSqrtvarnt(spot, strike, r, sigma);
  double mu_t = (gamma*sqrtvarnt + spot*eat)/strike;
  double atx = ( spot*eat*exp(x*sigma) - strike*(mu_t + si
    gma*x) + strike*sigma*( (1.0 - 0.5*t)*x ) );
  double btx = strike*sigma*sqrt(1.0/3.0 - (1.0 - 0.5*t)*(1
    .0 - 0.5*t)*t);

  return (atx*cdf_nor(atx/btx) + btx*pnl_normal_density(atx
    /btx));
}

double f_uppervw(double spot, double strike, double r,
    double sigma, double v, double w)
{

  return 2*v*pnl_normal_density(w)*f_upper(spot, strike, r,
     sigma, v*v, w*v);
}


static int ThompsonUp_FixedAsian(double pseudo_stock,
    double pseudo_strike,NumFunc_2  *po,double t,double r,double div
    id,double sigma,double *ptprice,double *ptdelta)
{
  int i,j;
  double sum,sum_delta,inc,new_r,new_sigma;
  double ascissev[NPOINTS+1],ascissew[NPOINTS+1],pesiv[NPOI
    NTS+1],pesiw[NPOINTS+1];
  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 and the delta  */
gauleg(0.0, 1.0,ascissev, pesiv, NPOINTS);
gauleg(-9.0, 10.0,ascissew, pesiw, NPOINTS);
sum=0.0;
sum_delta=0.;

for (j=1;j<=NPOINTS;j++) {
  for (i=1;i<=NPOINTS;i++){
    sum += pesiv[j]*pesiw[i]* f_uppervw(pseudo_stock, ps
  eudo_strike, new_r, new_sigma, ascissev[j], ascissew[i]);
    sum_delta +=pesiv[j]*pesiw[i]* f_uppervw(pseudo_stock
  *(1+inc), pseudo_strike, new_r, new_sigma, ascissev[j],
  ascissew[i]);
  }
}

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

/*  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)*(exp((r-divid)*t)-1.)/(t*(r-divid));
/*Delta for call option*/
Dlt=(exp(-r*t)*sum_delta-CTtK)/(pseudo_stock*inc);

/*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_ThompsonUp)(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_PDOUB
    LE;


  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[0].Val.V_PDOUBLE)/(ptOpt->Maturity.Val.V_
    DATE-(ptOpt->PathDep.Val.V_NUMFUNC_2)->Par[0].Val.V_PDOUB
    LE);
      pseudo_spot=(1.-time_spent)*ptMod->S0.Val.V_PDOUBLE;
      pseudo_strike=(ptOpt->PayOff.Val.V_NUMFUNC_2)->Par[0]
```

```
    .Val.V_PDOUBLE-time_spent*(ptOpt->PathDep.Val.V_NUMFUNC_2)
    ->Par[4].Val.V_PDOUBLE;

      if (pseudo_strike<=0.){
  Fprintf(TOSCREEN,"ANALYTIC FORMULA{n{n{n");
  return_value=Analytic_KemnaVorst(pseudo_spot,pseudo_stri
    ke,time_spent,ptOpt->PayOff.Val.V_NUMFUNC_2,ptOpt->Maturit
    y.Val.V_DATE-ptMod->T.Val.V_DATE,r,divid,&(Met->Res[0].Val.
    V_DOUBLE),&(Met->Res[1].Val.V_DOUBLE));
      }
      else
  return_value= ThompsonUp_FixedAsian(pseudo_spot,pseudo_
    strike,ptOpt->PayOff.Val.V_NUMFUNC_2,ptOpt->Maturity.Val.V_DA
    TE-ptMod->T.Val.V_DATE,r,divid,ptMod->Sigma.Val.V_PDOUBLE,&(
    Met->Res[0].Val.V_DOUBLE),&(Met->Res[1].Val.V_DOUBLE));
    }

  return return_value;
}

static int CHK_OPT(AP_FixedAsian_ThompsonUp)(void *Opt, voi
    d *Mod)
{
  if ( (strcmp(((Option*)Opt)->Name,"AsianCallFixedEuro")==
    0) || (strcmp( ((Option*)Opt)->Name,"AsianPutFixedEuro")==
    0) )
    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_ThompsonUp)=
```

```
{
  "AP_FixedAsian_ThompsonUp",
  {{" ",PREMIA_NULLTYPE,{0},FORBID}},
  CALC(AP_FixedAsian_ThompsonUp),
  {{"Price",DOUBLE,{100},FORBID},{"Delta",DOUBLE,{100},FORB
    ID} ,{" ",PREMIA_NULLTYPE,{0},FORBID}},
  CHK_OPT(AP_FixedAsian_ThompsonUp),
  CHK_ok,
  MET(Init)
};


#undef NPOINTS
```

References