

[Help](#)

```
#include "bs1d_pad.h"

/*Computation of Laplace transform*/
double fnRf_3_as(fcomplex a, double b, double c)
{
    long j;
    double pas,pr,Rfs;
    double wm,wa;
    fcomplex m,mu,som,der,gamm;

    pas=0.5/(b*1000.0);
    pr=0.0;
    mu=Csqrt(Cadd(Complex(c*c,0.0),RCmul(2.0,a)));
    som=Complex(0.0,0.0);

    /* Integral Computation */
    /* Rieman sums */
    for(j=1;j<1000;j++)
    {
        pr=pr+pas;
        wm=pas*exp(-pr+((mu.r-c)*0.5-2.)*log(pr)+((
            mu.r+c)*0.5+1.)*log(1.-2.*b*pr));
        wa=mu.i*0.5*log(pr-2.*b*pr*pr);
        som=Cadd(som,Complex(wm*cos(wa),wm*sin(wa)))
        ;
    }

    der=Complex(0.5*(mu.r-c)-1.,0.5*mu.i);
    gamm=Cgamma(der);
    gamm=Cmul(a,gamm);
    gamm=Cmul(gamm,Complex(a.r-(2.+2.*c),a.i));
    m=Cdiv(som,gamm);
    Rfs=m.r;

    return Rfs;
}
```

```

static int Laplace_FixedAsian(double pseudo_stock
    ,double pseudo_strike,NumFunc_2 *po,double t,
    double r,double divid,double sigma,double *ptprice,
    double *ptdelta)
{
    int N=15,M=11;
    int i;
    double xx,y,hh,sum,sum1,Avg,Avg1,Fun,Fun1,j,S[1
        1],U,tt,d,Q[11],pp;

    /* Inversion Variables*/
    fcomplex a;
    double St1,St2;
    double sigma2;
    double v,h,q,p,CTtK,PTtK,Dlt,Plt;
    double A;

    /*Inversion parameters*/
    A=19.1;
    pp=1.e-8;
    St1=pseudo_stock;
    St2=St1*(1.+pp);

    sigma2=sigma*sigma;
    v=2.0*(r-divid)/sigma2-1.;
    h=sigma2*t*0.25;
    q=sigma2*(pseudo_strike*t)/(4.0*St1);
    p=sigma2*(pseudo_strike*t)/(4.0*St2);

    /* INVERSION */
    tt=h;
    xx=A/(2*tt);
    a=Complex(xx, 0.0);
    sum=0.5*fnRf_3_as(a,q,v);
    sum1=0.5*fnRf_3_as(a,p,v);
    hh=PI/tt;

    /* Computation of S[1]=s(n) which approximate
        f(t) */
    for(i=1;i<=N;i++)
    {

```

```

y=i*hh;
a=Complex(xx,y);
j=pow(-1.0,(double)i);
sum = sum + j*fnRf_3_as(a,q,v);
sum1= sum1+j*fnRf_3_as(a,p,v);
}

S[0]=sum;
Q[0]=sum1;

/* End of Inversion */

/* Computation of s(n+p) p<=M+1 for Euler app
romations */

for(i=1;i<=M;i++)
{
    y=(N+i-1)*hh;
    a=Complex(xx,y);
    j=pow(-1.0,(double)N+i-1);
    S[i]=S[i-1]+j*fnRf_3_as(a,q,v);
    Q[i]=Q[i-1]+j*fnRf_3_as(a,p,v);
}

/* Computation of Euler appromations */

Avg=0.0;
Avg1=0.0;
for(i=0;i<=M;i++)
{
    Avg=Avg+Cnp(M,i)*S[i];
    Avg1=Avg1+Cnp(M,i)*Q[i];
}
d=pow(2.0,(double)M);
U=exp(A/2.)/tt;

/*f(t) value*/
Fun=U*Avg/d;
Fun1=U*Avg1/d;

/* Call Price */

```

```

CTtK=exp(-r*t)*4.0*St1*Fun/(t*sigma2);

/* Put Price from Parity*/
if(r==divid)
PTtK=CTtK+pseudo_strike*exp(-r*t)-St1*exp(-r*
t);
else
PTtK=CTtK+pseudo_strike*exp(-r*t)-St1*exp(-r*t)
*(exp((r-divid)*t)-1)/(t*(r-divid));

/*Delta for call option*/
Dlt=(exp(-r*t)*4.0*St2*Fun1/(t*sigma2)-exp(-r*
t)*4.0*St1*Fun/(t*sigma2))/(St1*pp);

/*Delta for put option*/
if(r==divid)
Plt=Dlt-exp(-r*t);
else
Plt=Dlt-exp(-r*t)*(exp((r-divid)*t)-1)/(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_Laplace)(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[0].Val.V_PDOUBLE)/(ptOpt-
    >Maturity.Val.V_DATE-(ptOpt->PathDep.Val.V_
    NUMFUNC_2)->Par[0].Val.V_PDOUBLE);
pseudo_spot=(1.-time_spent)*ptMod->S0.Val.V_PD
    OUBLE;
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_strike,time_spent,ptOpt->PayOff.Val.V_
    NUMFUNC_2,ptOpt->Maturity.Val.V_DATE-ptMod->T.Val.V_DA
    TE,r,divid,&(Met->Res[0].Val.V_DOUBLE),&(Met->Res[
    1].Val.V_DOUBLE));

```

```

    }
    else
        return_value= Laplace_FixedAsian(pseudo_spot,
            pseudo_strike,ptOpt->PayOff.Val.V_NUMFUNC_2,pt
            Opt->Maturity.Val.V_DATE-ptMod->T.Val.V_DATE,r,div
            id,ptMod->Sigma.Val.V_PDOUBLE,&(Met->Res[0].Val.
            V_DOUBLE),&(Met->Res[1].Val.V_DOUBLE));
    }

    return return_value;
}

int CHK_OPT(AP_FixedAsian_Laplace)(void *Opt, vo
    id *Mod)
{
    if ( (strcmp(((Option*)Opt)->Name,"
        AsianCallFixedEuro")==0) || (strcmp( ((Option*)Opt)->Name,"
        AsianPutFixedEuro")==0) )
        return OK;
    return WRONG;
}

static int MET(Init)(PricingMethod *Met)
{
    return OK;
}

PricingMethod MET(AP_FixedAsian_Laplace)=
{
    "AP_FixedAsian_Laplace",
    {" ",END,0,FORBID},
    CALC(AP_FixedAsian_Laplace),
    {{"Price",DOUBLE,100,FORBID},{ "Delta",DOUBLE,10
        0,FORBID} ,{" ",END,0,FORBID}},
    CHK_OPT(AP_FixedAsian_Laplace),
    CHK_ok,
    MET(Init)
};

```

References