

## [Help](#)

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

#define NPOINTS_FUSAITAGL 100

/*Computation the double(Mellin+Laplace) transform of the density of arithmetic
static dcomplex mellintransform(dcomplex l, dcomplex n, double sg, double r)
{

    dcomplex mu, nterm1, nterm2, nterm3, dterm1, dterm2;
    dcomplex num, den, cv, cost;
    double v;

    v = 2 * r / (sg * sg) - 1.0;
    cv = Complex(v, 0.0);
    mu = Csqrt(Cadd(Complex(v * v, 0), RCmul(2.0, 1)));

    cost = RCmul(log(2.0 / (sg * sg)), n);

    nterm1 = Clgamma(Cadd(n, CONE));
    nterm2 = Clgamma(Cadd(RCmul(0.5, Cadd(mu, cv)), CONE));
    nterm3 = Clgamma(Csub(RCmul(0.5, Csub(mu, cv)), n));
    num = Cadd(Cadd(nterm1, nterm2), nterm3);

    dterm1 = Clgamma(RCmul(0.5, Csub(mu, cv)));
    dterm2 = Clgamma(Cadd(Cadd(RCmul(0.5, Cadd(mu, cv)), CONE), n));

    den = Cadd(dterm1, dterm2);

    return Cdiv(Cexp(Cadd(Csub(num, den), cost)), 1);
}

/*We use the Cauchy Gourat theorem to compute the derivatives of the double(Mell
static dcomplex dermellin(dcomplex l, double sg, double r, int nummom)
{
    dcomplex term, cv, mu;
    int i;
```

```

double r0, sumr, sumi/*,x[NPOINTS_FUSAITAGL+1],w[NPOINTS_FUSAITAGL+1]*/;
double v;
double *x, *w;

x = malloc((NPOINTS_FUSAITAGL + 1) * sizeof(double));
w = malloc((NPOINTS_FUSAITAGL + 1) * sizeof(double));

sumr = 0.0;
sumi = 0.0;

gauleg(0, 2 * M_PI, x, w, NPOINTS_FUSAITAGL);

v = 2 * r / (sg * sg) - 1.0;
cv = Complex(v, 0.0);
mu = Csqrt(Cadd(Complex(v * v, 0), RCmul(2.0, 1)));
r0 = Creal(RCmul(0.5, Csub(mu, cv)));
if (r0 > 1.0) r0 = 0.25;

for (i = 1; i <= NPOINTS_FUSAITAGL; i++)
{
    term = RCmul(pow(r0, nummom), Cexp(Complex(0.0, nummom * x[i])));
    sumr += w[i] * Creal(Cdiv(mellintransform(1, RCmul(r0, Cexp(Complex(0.0, x[i] * nummom))), term)));
    sumi += w[i] * Cimag(Cdiv(mellintransform(1, RCmul(r0, Cexp(Complex(0.0, x[i] * nummom))), term)));
}

free(x);
free(w);

return Complex(exp(factln(nummom)) * sumr / (2.0 * M_PI), exp(factln(nummom)) * sumi / (2.0 * M_PI));
}

/*Use the Abate-Whitt for numerical inversion of the Laplace transform*/
static double SumAW(double expiry,
                    double sg, double r, double aa, int terms, int totterms, int *termsleft)
{
    int k;
    double h = sg * sg * expiry / 4.0;
    double Eulero;

```

```

dcomplex term;
dcomplex sum;
double *sum_r;
sum_r = malloc((totterms - terms + 2) * sizeof(double));
sum = Complex(0.0, 0.0);
Eulero = 0.0;
sum = RCmul(1.0 / 2.0, dermellin(Complex(aa / (2.0 * h), 0), sg, r, nummoment))
for (k = 1; k <= totterms; k++)
{
    term = RCmul(PNL_ALTERNATE(k) , dermellin(Complex(aa / (2.0 * h) , k * M_P

    sum = Cadd(term, sum);

    if (terms <= k) sum_r[k - terms + 1] = sum.r;
}

for (k = 0; k <= totterms - terms; k++)
{
    Eulero = Eulero + bico(totterms - terms, k) * pow(2.0, -(totterms - terms)
}
free(sum_r);
return exp(aa / 2.0) * Eulero / h;

}

/*We obtain the logarithmic moments of the average*/
static double MomentiLnAbWh(double expiry, double sg, double r, double aa, int t
{
    double inv = SumAW(expiry, sg, r, aa, terms, totterms, nummom);

    return inv;
}

static int FusaiTagliani_FixedAsian(double pseudo_stock, double pseudo_strike, N
{
    int i;

    double sum = 0.0, sum_delta = 0.;
    /* double area =0.0;*/
    int nnodi = NPOINTS_FUSAITAGL;

```

```

double CTtK, PTtK, Dlt, Plt;
double k2, k3, k4, m1, m2, m3, m4;
double k2a, k3a, k4a, m1a, m2a, m3a, m4a, var, m;
double term1, term2, term3, term4, edgedens ;
double aa;
int terms, totterms;
double *x, *w;
/*Set parameters for Laplace inversion*/
aa = 18.4;
terms = 15;
totterms = 25;

x = malloc((NPOINTS_FUSAITAGL + 1) * sizeof(double));
w = malloc((NPOINTS_FUSAITAGL + 1) * sizeof(double));

/*Computation of the first four logarithmic moments*/
m1 = MomentiLnAbWh(t, sigma, r - divid, aa, terms, totterms, 1);
m2 = MomentiLnAbWh(t, sigma, r - divid, aa, terms, totterms, 2);
m3 = MomentiLnAbWh(t, sigma, r - divid, aa, terms, totterms, 3);
m4 = MomentiLnAbWh(t, sigma, r - divid, aa, terms, totterms, 4);

/*Fit the parameters m,var of normal density*/
var = m2 - m1 * m1;
m = m1;

/*Computation of the cumulants of the logarithm of the arithmetic average*/
k2 = m2 - m1 * m1;
k3 = m3 - 3 * m1 * m2 + 3 * m2 * m1 * m1 - 3 * m1 * m1 * m1;
k4 = m4 - 4 * m3 * m1 - 3 * m2 * m2 + 12 * m2 * m1 * m1 - 6 * m1 * m1 * m1 * m1;
/*k4 = m4 - 4 * m3 * m1 + 6 * m2 * m1 * m1 - 3 * m1 * m1 * m1 * m1 - 3 * k2 * m1;

/*Edgeworth Adjustment : Computation of theoretical moments of the
normal density*/
m1a = m;
m2a = m2;
m3a = m * m * m + 3 * m * var;
m4a = m * m * m * m + 6 * m * m * var + 3 * var * var;

/*Edgeworth Adjustment : Computation of theoretical cumulants of the
normal density*/

```

```

k2a = m2a - m1a * m1a ;
k3a = m3a - 3 * m1a * m2a + 3 * m2a * m1a * m1a - 3 * m1a * m1a * m1a;
/*k4a = m4a - 4 * m3a * m1a + 6 * m2a * m1a * m1a - 3 * m1a * m1a * m1a * m1a - 3
k4a = m4a - 4 * m3a * m1a - 3 * m2a * m2a + 12 * m2a * m1a * m1a - 6 * m1a * m

/*Integrate, using the Laguerre quadrature, for obtaining the call price */
gauleg(log(pseudo_strike * t / pseudo_stock), log(pseudo_strike * t / pseudo_s
sum = 0.0;
sum_delta = 0.;
for (i = 1; i <= NPOINTS_FUSAITAGL; i++)
{

    /*Density construction using Edgeworth Expansion*/
    term1 = Normdens(x[i], m, pow(var, 0.5));
    term2 = (k2 - k2a) * Der2Normdens(x[i], m, pow(var, 0.5)) / 2.;
    term3 = -(k3 - k3a) * Der3Normdens(x[i], m, pow(var, 0.5)) / 6.;
    term4 = ((k4 - k4a) + 3 * (k2 - k2a)) * Der4Normdens(x[i], m, pow(var, 0.5
    edgedens = term1 + term2 + term3 + term4;

    /*Integration with respect to payoff for obtaining the call price
    and delta*/
    sum += w[i] * (exp(x[i]) * pseudo_stock / t - pseudo_strike) * edgedens;
    sum_delta += w[i] * exp(x[i]) / t * edgedens;
}

/* 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) * (ex

/*Delta for call option*/
Dlt = exp(-r * t) * sum_delta;

/*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;

free(x);
free(w);
return OK;
}

int CALC(AP_FixedAsian_FusaiTagliani)(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_DATE) * ptMod->S0.Val.V_PDOUBLE;
    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 * ptMod->S0.Val.V_PDOUBLE;

    if (pseudo_strike <= 0.)
    {
        Fprintf(TOSCREEN, "ANALYTIC FORMULA\ n\ n\ n");
        return_value = Analytic_KemnaVorst(pseudo_spot, pseudo_strike, time_spent);
    }
    else
        return_value = FusaiTagliani_FixedAsian(pseudo_spot, pseudo_strike, ptOpt);
}

return return_value;
}

static int CHK_OPT(AP_FixedAsian_FusaiTagliani)(void *Opt, void *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_FusaiTagliani) =
{
    "AP_FixedAsian_FusaiTagliani",
    {" ", PREMIA_NULLTYPE, {0}, FORBID}},
    CALC(AP_FixedAsian_FusaiTagliani),
    {"Price", DOUBLE, {100}, FORBID}, {"Delta", DOUBLE, {100}, FORBID} , {" ", PREMIA_NULLTYPE, {0}, FORBID}}
}

```

```
    CHK_OPT(AP_FixedAsian_FusaiTagliani),  
    CHK_ok,  
    MET(Init)  
};  
#undef NPOINTS_FUSAITAGL
```