

## [Help](#)

```
#include <stdlib.h>
#include "
href../../../../mod/bs1d/bs1d_pad/bs1d_pad_h_src.pdfbs1d_pad.h"
#include "
href../../../../common/math/ap_fusai_levy/QDiscreteAsian_h_src.pdfmath/ap_fusai_levy
#include "
href../../../../common/math/ap_fusai_levy/nrutil_h_src.pdfmath/ap_fusai_levy/nrutil

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2008+2) //The "#els
static int CHK_OPT(AP_FixedAsian_FusaiMeucci)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
int CALC(AP_FixedAsian_FusaiMeucci)(void *Opt, void *Mod, PricingMethod *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
#else

static int FusaiMeucci_FixedAsian(double pseudo_stock, double pseudo_strike, Num
{
    double CTtK, PTtK, Dlt, Plt;
    double lowlim = 10., uplim = 15.;
    long int nfft = 65536;
    double *price, *solution, delta;

    price = dvector(0, M - 1);
    solution = dvector(0, M - 1);

    /* Call Price */
    CTtK = Asian_BS_FusaiMeucci(pseudo_stock, pseudo_strike, t, r, divid, sigma, N

    /* 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 = 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_dvector(price, 0, M - 1);
free_dvector(solution, 0, M - 1);

return OK;
}

int CALC(AP_FixedAsian_FusaiMeucci)(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);
    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 = FusaiMeucci_FixedAsian(pseudo_spot, pseudo_strike, ptOpt->T.Val.V_DATE, ptMod->S0.Val.V_PDOUBLE, ptMod->S0.Val.V_INT2, ptMod->S0.Val.V_INT1);
    }

    return return_value;
}

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

#endif //PremiaCurrentVersion
static int MET(Init)(PricingMethod *Met, Option *Opt)
{
    if (Met->init == 0)
    {
        Met->init = 1;
        Met->Par[0].Val.V_INT2 = 52;
        Met->Par[1].Val.V_INT2 = 3000;
    }
}

```

```

    return OK;
}

PricingMethod MET(AP_FixedAsian_FusaiMeucci) =
{
    "AP_FixedAsian_FusaiMeucci",
    { {"Nb.of Monitoring Dates", INT2, {2000}, ALLOW },
      {"Nb.of Integration Points ", INT2, {1000}, ALLOW},
      {" ", PREMIA_NULLTYPE, {0}, FORBID}
    },
    CALC(AP_FixedAsian_FusaiMeucci),
    {{"Price", DOUBLE, {100}, FORBID}, {"Delta", DOUBLE, {100}, FORBID} , {" ", PR
    CHK_OPT(AP_FixedAsian_FusaiMeucci),
    CHK_ok,
    MET(Init)
};

```