

[Help](#)

```
#include "bs1d_pad.h"

static int MilevskyPosner_FixedAsian(double pseu
    do_stock,double pseudo_strike,NumFunc_2 *po,
    double t,double r,double divid,double sigma,double *pt
    price,double *ptdelta)
{
    int i;
    double x[NGAMMA+1],w[NGAMMA+1];
    double m1,m2,a,b,sum,sum_delta,k;
    double CTtK,PTtK,Dlt,Plt;
    double new_stock,new_strike, new_r, new_sigma;

    /*Scaling of the parameters*/
    new_stock=1.;
    new_strike=pseudo_strike/pseudo_stock;
    new_r=(r-divid)*t;
    new_sigma=sigma*sqrt(t);

    /*Computation of the first two moments*/
    m1=Moments(1,new_r,new_sigma,1)*new_stock;
    m2=Moments(2,new_r,new_sigma,1)*new_stock*new_
        stock;

    /*Fit the parameters a,b of reciprocal gamma*/
    a=(2.*m2-m1*m1)/(m2-m1*m1);
    b=(m2-m1*m1)/(m2*m1);

    /*Integrate, using the Laguerre quadrature, th
        e payoff function of Put option */
    k=new_strike/new_stock;
    gauleg(0, k, x, w, NGAMMA);
    sum=0.;
    sum_delta=0.;
    for (i=1;i<=NGAMMA;i++) {
        sum += w[i]*(new_strike - x[i]*new_stock)*gam
            madensity(1.0/x[i], a, b)/(x[i]*x[i]);
        sum_delta += w[i]*(- x[i]*new_stock)*gammaden
            sity(1.0/x[i], a, b)/(x[i]*x[i]);
    }
}
```

```

/* Put Price*/
PTtK=pseudo_stock*exp(-r*t)*sum;

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

/*Delta for put option*/
Plt=exp(-r*t)*sum_delta;

/*Delta for call option*/
if(r==divid)
Dlt=Plt+exp(-r*t);
else
Dlt=Plt+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_MilevskyPosner)(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->Pat
hDep.Val.V_NUMFUNC_2)->Par[0].Val.V_PDOUBLE)/(pt
Opt->Maturity.Val.V_DATE-(ptOpt->PathDep.Val.V_
NUMFUNC_2)->Par[0].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*(ptOpt->PathD
ep.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_sp
ot,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=MilevskyPosner_FixedAsian(pseu
            do_spot,pseudo_strike,ptOpt->PayOff.Val.V_NUMFUNC_
            2,ptOpt->Maturity.Val.V_DATE-ptMod->T.Val.V_DA
            TE,r,divid,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_MilevskyPosner)(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)
{
    return OK;
}

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

#undef EPS

```

References