

[Help](#)

```
#include "bs1d_pad.h"

static int FSG_Asian(int type_asian,int am,
    double x,double y_1,double y_2,double K,NumFunc_2 *p,
    double T,double r,double divid,double sigma,int N,int
    oneoverrho,double *ptprice,double *ptdelta)
{
    double **C_n,**C_n_minus_one;
    int n,j,k;
    double h,u,d,pu,pd,a1;
    double price;
    double dY,dZ;
    int kfloor_p,kfloor_m;
    double epsilon_p,epsilon_m,psi_p,psi_m,expdy,
    asian_value,spot_value,expdz,delta_factor;

    /*Parameter for the pathdep discretization*/
    oneoverrho *= (int)floor(sqrt(N));

    /*Memory Allocation*/
    C_n = (double* *)malloc(sizeof(double *)*(2*
    N+1));
    if (C_n == NULL)
        return MEMORY_ALLOCATION_FAILURE;
    C_n_minus_one = (double* *)malloc(sizeof(
    double *)*(2*N+1));
    if (C_n_minus_one == NULL)
        return MEMORY_ALLOCATION_FAILURE;

    for(j=-N;j<=N;j++){
        C_n[N+j] = (double *)malloc(sizeof(
        double)*(2*N*oneoverrho+1));
        if (C_n[N+j] == NULL)
            return MEMORY_ALLOCATION_FAILURE;
        C_n_minus_one[N+j] = (double *)malloc(siz
        eof(double)*(2*N*oneoverrho+1));
        if (C_n_minus_one[N+j] == NULL)
            return MEMORY_ALLOCATION_FAILURE;
    }
}
```

```

/* Up and Down factors */
h = T / (double)N;
dZ = sigma*sqrt(h);
dY = dZ / (double)oneoverrho;
expdy = exp(dY);
expdz = exp(dZ);

/* Discrete risk-neutral probability */
a1 = exp(h*(r-divid));
u = exp(sigma*sqrt(h));
d = 1. / u;

pu = (a1 - d) / (u-d);
pd = 1. - pu;
pu *= exp(-r*h) ;
pd *= exp(-r*h) ;

/*Ratio for the delta*/
if (type_asian==1)
delta_factor=x;
else
delta_factor=y_1;

/*Intrinsic value initialisation and terminal
values*/
spot_value = x*exp(-(double)(N+1)*dZ);

for(j=-N;j<=N;j++)
{
    spot_value*=expdz;
    asian_value=y_1*exp(-(double)(N*oneo
verrho+1)*dY);

    for(k=-N*oneoverrho;k<=N*oneoverrho;k
++)
    {
        asian_value*=expdy;
        C_n[N+j][N*oneoverrho+k] =(p-
>Compute)(p->Par,spot_value,y_2+asian_value);
    }
}

```

```

/*Backward resolution*/
for(n=N-1;n>0;n--)
{
    spot_value=x*exp(-(double)(n+1)*dZ);

    for(j=-n;j<=n;j++)
    {

        spot_value*=expdz;
        asian_value=y_1*exp(-(double)
(n*oneoverrho+1)*dY);

        for(k=-n*oneoverrho;k<=n*oneo
verrho;k++)
        {
            asian_value*=expdy;

            psi_p      = ( (n+1)*
asian_value / y_1 + spot_value * expdz / x) / (n+2);
            psi_m      = ( (n+1)*
asian_value / y_1 + spot_value / (x* expdz)) / (n+2);

            kfloor_p   = (int)floo
r(log(psi_p) / dY);
            kfloor_m   = (int)floo
r(log(psi_m) / dY);

            epsilon_p = (psi_p *
exp(-kfloor_p * dY) -1.0) / (expdy - 1.0);
            epsilon_m = (psi_m*
exp(-kfloor_m * dY) -1.0 ) / (expdy - 1.0);

            if((N - n) % 2 == 1)
            {
                price = pu *
( (1. - epsilon_p) * C_n[N+j+1][N*oneoverrho+k
floor_p] + epsilon_p * C_n[N+j+1][N*oneoverrho+k
floor_p+1] ) + pd * ( (1. - epsilon_m) * C_n[N+j-
1][N*oneoverrho+kfloor_m] + epsilon_m * C_n[N+j-
1][N*oneoverrho+kfloor_m+1] );

```

```

    } else {

        price = pu *
        ( (1. - epsilon_p) * C_n_minus_one[N+j+1][N*oneoverrho+kfloor_p] + epsilon_p * C_n_minus_one[N+j+1][N*oneoverrho+kfloor_p+1] ) + pd * ( (1. - epsilon_m) * C_n_minus_one[N+j-1][N*oneoverrho+kfloor_m] + epsilon_m * C_n_minus_one[N+j-1][N*oneoverrho+kfloor_m+1] );

    }
    if(am)
        price=MAX(price,
        (p->Compute)(p->Par,spot_value,y_2+asian_value))
    ;
    if((N - n) % 2 == 1)
        C_n_minus_one[j+
N][k+N*oneoverrho] = price;
    else
        C_n[j+N][k+N*oneoverrho] = price;

    }

}

psi_p    = ( 1.0 + expdz) / 2.0;
psi_m    = ( 1.0 + 1.0 / expdz) / 2.0;

kfloor_p = (int)floor(log(psi_p) / dY);
kfloor_m = (int)floor(log(psi_m) / dY);

epsilon_p = (psi_p * exp(-kfloor_p * dY) - 1.0) / (expdy - 1.0);
epsilon_m = (psi_m * exp(-kfloor_m * dY) - 1.0) / (expdy - 1.0);

/* First Step*/

```

```

if(N % 2 == 1)
{
    *ptdelta = ( ( (1. - epsilon_p) * C_n[
N+1][N*oneoverrho+kfloor_p] + epsilon_p * C_n[N+1]
][N*oneoverrho+kfloor_p+1] ) - ( (1. - epsilon_
m) * C_n[N-1][N*oneoverrho+kfloor_m] + epsilon_m
* C_n[N-1][N*oneoverrho+kfloor_m+1] ) ) / (( u -
d ) * delta_factor );
    *ptprice = pu * ( (1. - epsilon_p) * C_
n[N+1][N*oneoverrho+kfloor_p] + epsilon_p * C_n[
N+1][N*oneoverrho+kfloor_p+1] ) + pd * ( (1. - ep
silon_m) * C_n[N-1][N*oneoverrho+kfloor_m] + epsi
lon_m * C_n[N-1][N*oneoverrho+kfloor_m+1] );
}
else
{
    *ptdelta = ( ( (1. - epsilon_p) * C_n_
minus_one[N+1][N*oneoverrho+kfloor_p] + epsilon_
p * C_n_minus_one[N+1][N*oneoverrho+kfloor_p+1] )
- ( (1. - epsilon_m) * C_n_minus_one[N-1][N*on
eoverrho+kfloor_m] + epsilon_m * C_n_minus_one[N-
1][N*oneoverrho+kfloor_m+1] ) ) / ( ( u - d ) *
delta_factor );
    *ptprice = pu * ( (1. - epsilon_p) * C_
n_minus_one[N+1][N*oneoverrho+kfloor_p] + epsilon
_p * C_n_minus_one[N+1][N*oneoverrho+kfloor_p+1]
) + pd * ( (1. - epsilon_m) * C_n_minus_one[N-1]
[N*oneoverrho+kfloor_m] + epsilon_m * C_n_minus
_one[N-1][N*oneoverrho+kfloor_m+1] );
}
if(am)
{
    *ptprice=MAX(*ptprice, (p->Compute)(p->
Par,x,y_2+asian_value));
}

/* Memory Desallocation */
for(j=-N;j<=N;j++) {
    free(C_n[N+j]);
    free(C_n_minus_one[N+j]);
}

```

```

    free(C_n);
    free(C_n_minus_one);

    return OK;
}

int CALC(TR\_Aasian\_FSG)(void *Opt,void *Mod,PricingMethod *Met)
{
    TYPEOPT* ptOpt=( TYPEOPT*)Opt;
    TYPEMOD* ptMod=( TYPEMOD*)Mod;
    double r,divid,time_spent,asian_spot,pseudo_
spot,T_0,t_0,T;
    int return_value,type_asian;

    r=log(1.+ptMod->R.Val.V_DOUBLE/100.);
    divid=log(1.+ptMod->Divid.Val.V_DOUBLE/100.);

    T = ptOpt->Maturity.Val.V_DATE;
    t_0 = (ptOpt->PathDep.Val.V_NUMFUNC_2)->Par[0]
.Val.V_PDOUBLE;
    T_0 = ptMod->T.Val.V_DATE;

    time_spent = (T_0 - t_0 ) /(T - t_0);
    asian_spot = (ptOpt->PathDep.Val.V_NUMFUNC_2)
->Par[4].Val.V_PDOUBLE*time_spent;
    pseudo_spot = (1. - time_spent)*ptMod->S0.Val
.V_PDOUBLE;

    if( T_0 < t_0)
    {
        return_value = 0;
    } else {
        if (((ptOpt->PayOff.Val.V_NUMFUNC_2)->Compute==Call_StrikeSpot2)||
            ((ptOpt->PayOff.Val.V_NUMFUNC_2)->Compute==Put_StrikeSpot2))
            /*Floating Case*/

```

```

    type_asian=1;
    else type_asian=0;

    return_value=FSG_Aasian(type_asian,pt
    Opt->EuOrAm.Val.V_BOOL,ptMod->S0.Val.V_PDOUBLE,pseu
    do_spot,asian_spot, (ptOpt->PayOff.Val.V_NUMFUNC_2
    )->Par[0].Val.V_PDOUBLE,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->Par[0].
    Val.V_INT2,Met->Par[1].Val.V_INT2,&(Met->Res[0].
    Val.V_DOUBLE),&(Met->Res[1].Val.V_DOUBLE));
    }

    return return_value;
}

int CHK_OPT(TR_Aasian_FSG)(void *Opt, void *Mod)
{
    if ( (strcmp( ((Option*)Opt)->Name,"
    AsianCallFixedEuro")==0) || (strcmp( ((Option*)Opt)->Name,"
    AsianPutFixedEuro")==0) || (strcmp( ((Option*)Opt)->Name,"
    AsianCallFloatingEuro")==0) || (strcmp( ((Option*)Opt)->Name,"
    AsianPutFloatingEuro")==0) || (strcmp( ((Option*)Opt)->Name,"
    AsianCallFixedAmer")==0) || (strcmp( ((Option*)Opt)->Name,"
    AsianPutFixedAmer")==0) || (strcmp( ((Option*)Opt)->Name,"
    AsianCallFloatingAmer")==0) || (strcmp( ((Option*)Opt)->Name,"
    AsianPutFloatingAmer")==0))
        return OK;

    return WRONG;
}

static int MET(Init)(PricingMethod *Met)
{
    static int first=1;

    if (first)
    {
        Met->Par[0].Val.V_INT2=50;
        Met->Par[1].Val.V_INT2=2;
    }
}

```

```
        first=0;
    }

    return OK;
}

PricingMethod MET(TR\_Asian\_FSG) =
{
    "TR\_Asian\_FSG",
    {"StepNumber",INT2,100,ALLOW},{"Inverse of
    Rho",INT2,100,ALLOW},{" ",END,0,FORBID}},
    CALC(TR\_Asian\_FSG),
    {"Price",DOUBLE,100,FORBID},{"Delta",DOUBLE,
    100,FORBID} ,{" ",END,0,FORBID}},
    CHK_OPT(TR\_Asian\_FSG),
    CHK_tree,
    MET(Init)
};
```

## References