

Help

```

#include<stdlib.h>
#include<time.h>
#include"pnl/pnl_specfun.h"
#include "nig1d_pad.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
    (2011+2) //The "#else" part of the code will be freely available after the (year of creation of this file + 2)
static int CHK_OPT(AP_CernyKyriakou_NIG_FixedAsian)(void *
    Opt, void *Mod)
{
    return NONACTIVE;
}
int CALC(AP_CernyKyriakou_NIG_FixedAsian)(void*Opt,void *
    Mod,PricingMethod *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
#else
//Laplace transform of NIG process
static dcomplex NIG_laplace_transform(dcomplex u,double t,
    double alpha,double beta,double delta,double drift)
{
    dcomplex temp;

    temp=RCsub(delta*sqrt(alpha*alpha-beta*beta),RCmul(delta,
        Csqrt(RCsub(alpha*alpha,Cmul(RCadd(beta,u),RCadd(beta,u))))
        ));

    return Cexp(RCmul(t,Cadd(RCmul(drift,u),temp)));
}
//The generalized discrete Fourier transform (DFT) of a function f from
    m x onto u
static void DFT(dcomplex *a,double *x,int n,double *u,int
    m,dcomplex* b)
{
    int j,k;

    for(k=0;k<=m;k++)
    {

```

```

    b[k]=CZERO;
    for(j=1;j<=n;j++)
        b[k]=Cadd(Cmul(a[j],CIexp(x[j]*u[k])),b[k]);
    b[k]=CRmul(b[k],fabs(x[1]-x[0]));
}
}

int nig_ap_cernykyriakou_asianfixed(NumFunc_2*P,double S0,
    double T,double r,double divid,double sigma,double theta,double
    kappa,int n_points,double *ptprice)
{
    double *lambda,drift,*L,*L_bar,*u,*U,*U_bar,rho,u_max,*x
        ,*y,x_step,y_step,time_step;
    double u_step,*u_minus,*h,u_em,l_em,*a,*b,exp_X,alpha,
        beta,delta,K,coef;
    int i,k,i_up,i_down,x_nb,y_nb,u_nb,i_temp,i_x,test;
    dcomplex *q,*p,*P_vect,*phi,*temp;

    alpha=sqrt(theta*theta+sigma*sigma/kappa)/(sigma*sigma);
    beta=theta/(sigma*sigma);
    delta=sigma/sqrt(kappa);
    K=P->Par[0].Val.V_DOUBLE;
    rho=1e-6;
    time_step=T/n_points;
    u_nb=POW(2,10);
    x_nb=POW(2,10);
    y_nb=POW(2,10);
    i_x=0;
    coef=1;
    lambda=malloc((n_points+1)*sizeof(double));
    a=malloc((n_points+1)*sizeof(double));
    b=malloc((n_points+1)*sizeof(double));
    L=malloc((n_points+1)*sizeof(double));
    U=malloc((n_points+1)*sizeof(double));
    L_bar=malloc((n_points+1)*sizeof(double));
    U_bar=malloc((n_points+1)*sizeof(double));
    x=malloc((x_nb+1)*sizeof(double));
    y=malloc((y_nb+1)*sizeof(double));
    h=malloc((y_nb+1)*sizeof(double));
    p=malloc((y_nb+1)*sizeof(dcomplex));
    q=malloc((x_nb+1)*sizeof(dcomplex));

```

```

drift=(r-divid)-log(Creal(NIG_laplace_transform(Complex(
    1,0),1,alpha,beta,delta,0)));
u_max=5.;
while(Cabs(NIG_laplace_transform(Complex(0,u_max),time_s
    tep,alpha,beta,delta,drift))+Cabs(NIG_laplace_transform(
    Complex(0,-u_max),time_step,alpha,beta,delta,drift))>rho)
{
    u_max+=5.;
}
u_step=2*u_max/u_nb;
if(u_step>0.125)
{
    u_nb=trunc(16*u_max);
    u_nb+=PNL_IS_ODD(u_nb);
    u_step=2*u_max/u_nb;
}
u=malloc((u_nb+1)*sizeof(double));
u_minus=malloc((u_nb+1)*sizeof(double));
P_vect=malloc((u_nb+1)*sizeof(dcomplex));
phi=malloc((u_nb+1)*sizeof(dcomplex));
temp=malloc((u_nb+1)*sizeof(dcomplex));
for(i=0;i<=u_nb;i++)
{
    u[i]=-u_max+i*u_step;
    u_minus[i]=-u[i];
    phi[i]=NIG_laplace_transform(Complex(0,u[i]),time_step
    ,alpha,beta,delta,drift);
}
/*Call Case*/
lambda[0]=1./(n_points+1)-K/S0;
for(i=1;i<=n_points;i++)
    lambda[i]=1./(n_points+1);
do
{
    u_em=5*coef;
    l_em=-5*coef;
    ////////////////////////////////////////////
    ////////////////////////////////////////////
    a[n_points]=1;
    b[n_points]=lambda[0]*(lambda[0]>0);

```

```

    exp_X=Creal(NIG_laplace_transform(Complex(1,0),time_s
tep,alpha,beta,delta,drift));
    for(i=n_points;i>0;i--)
    {
        a[i-1]=a[i]*exp_X;
        b[i-1]=b[i]+a[i-1]*lambda[n_points-i+1];
    }
    ////////////////////////////////////////////
    ////////////////////////////////////////////
    L_bar[0]=log(lambda[n_points]);
    U_bar[0]=log(lambda[n_points]);
    for(i=1;i<n_points;i++)
    {
        L[i]=L_bar[i-1]+l_em;
        U[i]=U_bar[i-1]+u_em;
        L_bar[i]=log(exp(L[i])+lambda[n_points-i]);
        U_bar[i]=log(exp(U[i])+lambda[n_points-i]);
    }
    L[n_points]=L_bar[n_points-1]+l_em;
    U[n_points]=U_bar[n_points-1]+u_em;
    y_step=(U[n_points]-L[n_points])/y_nb;
    for(i=0;i<=y_nb;i++)
    {
        y[i]=L[n_points]+i*y_step;
        p[i]=Complex((exp(y[i])+lambda[0])*(exp(y[i])+lambd
a[0]>0),0);
    }
    coef*=0.95;
}
while(y[y_nb]>10);
for(k=n_points;k>1;k--)
{

    x_step=(U_bar[k-1]-L_bar[k-1])/x_nb;
    for(i=0;i<=x_nb;i++)
    {
        x[i]=L_bar[k-1]+i*x_step;
    }
    DFT(p,y,y_nb,u,u_nb,P_vect);

```

```

    for(i=0;i<=u_nb;i++)
    {
        temp[i]=Cmul(P_vect[i],Conj(phi[i]));
    }
    DFT(temp,u_minus,u_nb,x,x_nb,q);
    for(i=0;i<=x_nb;i++)
    {
        q[i]=CRdiv(q[i],2*M_PI);
    }
    ////////////////////////////////////////////
    ////////////////////////////////////////////
    y_step=(U[k-1]-L[k-1])/y_nb;
    for(i=0;i<=y_nb;i++)
    {
        y[i]=L[k-1]+i*y_step;
        h[i]=log(exp(y[i])+lambda[n_points-(k-1)]);

        test=0;
        if(h[i]<=x[0])
        {
            i_x=0;
            test=1;
            p[i]=Cadd(q[i_x],CRmul(CRdiv(Csub(q[i_x+1],q[i_x]),
x[i_x+1]-x[i_x]), (h[i]-x[i_x]))));
        }
        if(h[i]>=x[x_nb])
        {
            i_x=x_nb;
            test=1;
            p[i]=Cadd(q[i_x-1],CRmul(CRdiv(Csub(q[i_x],q[i_x-1
]), x[i_x]-x[i_x-1]), (h[i]-x[i_x-1]))));
        }
        if((h[i]>x[0]) && (h[i]<x[x_nb]))
        {
            i_down=x_nb/2;
            i_up=x_nb;
            if(x[i_down]>h[i])
            {
                i_down=0;
                i_up=x_nb/2;
            }
        }
    }

```

```

while(test==0)
{
    if(x[i_down+1]>h[i])
    {
        i_x=i_down;
        test=1;
    }
    else
    {
        i_temp=(i_up-i_down-1)/2+i_down;
        if(x[i_temp]>h[i])
        {
            i_up=i_temp;
            i_down=i_down+1;
        }
        else
        {
            i_down=i_temp*(i_temp>i_down)+(i_down+1)*(i_
temp<=i_down);
        }
    }
}
p[i]=Cadd(q[i_x],CRmul(CRdiv(Csub(q[i_x+1],q[i_x])
,x[i_x+1]-x[i_x]),(h[i]-x[i_x]))));
}
}
////////////////////////////////////
//////////
}
x_step=(U_bar[0]-L_bar[0])/x_nb;
for(i=0;i<=x_nb;i++)
{
    x[i]=L_bar[0]+i*x_step;
}
DFT(p,y,y_nb,u,u_nb,P_vect);
for(i=0;i<=u_nb;i++)
{
    temp[i]=Cmul(P_vect[i],Conj(phi[i]));
}
DFT(temp,u_minus,u_nb,x,x_nb,q);
for(i=0;i<=x_nb;i++)

```

```

    {
        q[i]=CRdiv(q[i],2*M_PI);
    }
    /*Call Case*/
    if((P->Compute)==&Call_OverSpot2)
    {
        *ptprice=S0*exp(-r*T)*Creal(q[0]);
    }
    /*Put case*/
    if((P->Compute)==&Put_OverSpot2)
    {
        if(r!=divid)
            *ptprice=S0*exp(-r*T)*Creal(q[0])+K*exp(-r*T)-S0*(
exp(-divid*T)-exp(-r*T))/((r-divid)*T);
        else
            *ptprice=S0*exp(-r*T)*Creal(q[0])-(S0-K)*exp(-r*T);
    }
    free(lambda);
    free(L);
    free(U);
    free(L_bar);
    free(U_bar);
    free(u);
    free(u_minus);
    free(x);
    free(y);
    free(h);
    free(p);
    free(P_vect);
    free(phi);
    free(temp);
    free(q);
    free(a);
    free(b);

    return OK;
}
int CALC(AP_CernyKyriakou_NIG_FixedAsian)(void*Opt,void *
Mod,PricingMethod *Met)
{
    TYPEOPT* ptOpt=(TYPEOPT*)Opt;

```

```

TYPEMOD* ptMod=(TYPEMOD*)Mod;
double r,divid;

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

return nig_ap_cernykyriakou_asianfixed(ptOpt->PayOff.Val
    .V_NUMFUNC_2,ptMod->SO.Val.V_PDOUBLE,ptOpt->Maturity.Val.
    V_DATE-ptMod->T.Val.V_DATE,r,divid,ptMod->Sigma.Val.V_PDOUB
    LE,ptMod->Theta.Val.V_DOUBLE,ptMod->Kappa.Val.V_SPDOUBLE,
    Met->Par[0].Val.V_PINT,&(Met->Res[0].Val.V_DOUBLE));
}

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

#endif //PremiaCurrentVersion
static int MET(Init)(PricingMethod *Met,Option *Mod)
{
    if ( Met->init == 0)
    {
        Met->init=1;
        Met->HelpFilenameHint ="    ap_cernykyriakou_nig_asianfixed";
        Met->Par[0].Val.V_PINT=10;
    }
    return OK;
}

PricingMethod MET(AP_CernyKyriakou_NIG_FixedAsian)=
{
    "AP_CernyKyriakou_NIG_FixedAsian",
    {"Number of discretization steps",LONG,{100},ALLOW},{ " "
        ,PREMIA_NULLTYPE,{0},FORBID}},
    CALC(AP_CernyKyriakou_NIG_FixedAsian),

```



```
{{"Price",DOUBLE,{100},FORBID},{ " ",PREMIA_NULLTYPE,{0},
  FORBID}},
CHK_OPT(AP_CernyKyriakou_NIG_FixedAsian),
CHK_ok,
MET(Init)
} ;
```

References