

## Help

```

#include <stdlib.h>
#include "nig1fact1d_std.h"

#include "pnl/pnl_vector.h"
#include "pnl/pnl_random.h"
#include "pnl/pnl_complex.h"
#include "pnl/pnl_integration.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
    (2011+2) //The "#else" part of the code will be freely av
    ailable after the (year of creation of this file + 2)
static int CHK_OPT(AP_GOR)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
int CALC(AP_GOR)(void*Opt,void *Mod,PricingMethod *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
#else

static dcomplex nig_laplace(dcomplex u,double *param)
{
    dcomplex temp1;

    temp1=RCmul(param[4],RCsub(sqrt(POW(param[2],2)-POW(
    param[3],2)),Csqrt(RCsub(POW(param[2],2),Cpow_real(CRadd(u,
    param[3]),2)))));
    temp1=Cadd(temp1,CRmul(u,param[5]));
    temp1=Cexp(CRmul(temp1,param[6]));

    return temp1;
}
static double H_func_real(double t,void *V)
{
    dcomplex x,c1,c2,H_func;
    double * param=(double *)V;
    c1=Complex(param[8],-param[9]);
    c2=Complex(param[8],param[9]);

```

```

x=Cadd(RCmul((1-t),c1),RCmul(t,c2));

H_func= Cdiv(Cmul(Cpow(Complex(param[0]/param[1],0),RCsub(
  b(1,x)),((Cpow_real(Csub(nig_laplace(x,param),CRdiv(RCmul(
  param[11],(Csub(nig_laplace(CRadd(x,1),param),RCmul(param[10]
  ,nig_laplace(x,param))))),param[12])),param[7])))),Cmul(x,
  CRsub(x,1))));

  return Creal(H_func)*param[1]*param[9]/M_PI;
}
static double Eps_func_real(double t,void *V)
{
  dcomplex x,c1,c2,Eps_func;
  double * param=(double *)V;
  c1=Complex(param[8],-param[9]);
  c2=Complex(param[8],param[9]);
  x=Cadd(RCmul(1-t,c1),RCmul(t,c2));

  Eps_func=Cdiv(Cmul(Cpow(Complex(param[0]/param[1],0),RCsub(
    sub(1,x)),( Cmul((CRdiv(Csub(nig_laplace(Cadd(x,CONE),
    param),RCmul(param[10],nig_laplace(x,param))),param[12])),Cpow_real(
    Csub(nig_laplace(x,param),CRdiv(RCmul(param[11],Csub(
    nig_laplace(CRadd(x,1),param),RCmul(param[10],nig_laplace(
    e(x,param))))),param[12])),param[7])))),Cmul(x,CRsub(x,1)))
  ;

  return Creal(Eps_func)*param[9]/M_PI;
}

static int ap_GouteOudjaneRusso(double K,double S0,
  double T,double r,double alpha,double beta,double delta,double
  mu,int n_points,double *ptprice, double *ptdelta)
{
  /*Computation of the parameters of the Follmer-Schweizer decomposition
  of the payoff f(S_T) where f is a function can be written with
  respect to the complex measure Pi(dz) (it is an European Call).*/
  //////////////////////////////////////
  /* H, Eps: Parameters of the FS decomposition*/

```

```

double R,B,*param,M1,M2,L1,g2;
int neval;
PnlFunc funcH, funcEps;

param=malloc(16*sizeof(double));
R=2;    //R has to be strictly bigger than 1 (real
part)
B=1000;
param[0]=K;
param[1]=S0;
param[2]=alpha;
param[3]=beta;
param[4]=delta;
param[5]=mu;
param[6]=T/n_points;
param[7]=n_points;
param[8]=R;
param[9]=B;
M1=Creal(nig_laplace(CONE,param));
M2=Creal(nig_laplace(Complex(2,0),param));
L1=M1-1;
g2=M2-M1*M1;
param[10]=M1;
param[11]=L1;
param[12]=g2;
funcH.function=H_func_real;
funcH.params=param;
funcEps.function=Eps_func_real;
funcEps.params=param;
neval=100000;

*ptprice=pnl_integration(&funcH,0,1,neval,"simpson");
*ptdelta=pnl_integration(&funcEps,0,1,neval,"simpson");

free(param);

return 0;
}

/*////////////////////////////////////*/

```

```

static int ap_russo(NumFunc_1 p,double S0, double T,
    double K,
                double alpha,double beta,double delta,
    double mu,
                double r,double sigma,double lambda,
                double *ptprice, double *ptdelta)
{
    int n_points=10;
    ap_GouteOudjaneRusso(K, S0, T, r, alpha, beta, delta,
    mu, n_points, ptprice, ptdelta);

    return OK;
}

//=====
=====
int CALC(AP_GOR)(void *Opt,void *Mod,PricingMethod *Met)
{
    TYPEOPT* ptOpt=( TYPEOPT*)Opt;
    TYPEMOD* ptMod=( TYPEMOD*)Mod;
    double r,strike;
    NumFunc_1 *p;
    int res;

    r=log(1.+ptMod->R.Val.V_DOUBLE/100.);
    p=ptOpt->PayOff.Val.V_NUMFUNC_1;
    strike=p->Par[0].Val.V_DOUBLE;

    res = ap_russo(*p,ptMod->S0.Val.V_PDOUBLE,ptOpt->Matu
    rity.Val.V_DATE-ptMod->T.Val.V_DATE,strike,ptMod->alpha.Val.
    V_PDOUBLE,ptMod->beta.Val.V_DOUBLE,ptMod->delta.Val.V_SPDOU
    BLE,ptMod->mu.Val.V_SPDOUBLE,
                r, ptMod->Sigma.Val.V_PDOUBLE,ptMod->lam
    bda.Val.V_PDOUBLE,
                &(Met->Res[0].Val.V_DOUBLE), &(Met->Res[
    1].Val.V_DOUBLE));
    return res;
}

```

```

static int CHK_OPT(AP_GOR)(void *Opt, void *Mod)
{
    if ((strcmp( ((Option*)Opt)->Name,"CallEuro")==0))
        return OK;

    return WRONG;
}
#endif //PremiaCurrentVersion

static int MET(Init)(PricingMethod *Met,Option *Opt)
{
    if ( Met->init == 0)
    {
        Met->init=1;
    }
    return OK;
}

PricingMethod MET(AP_GOR)=
{
    "AP_GOR",
    {{" ",PREMIA_NULLTYPE,{0},FORBID}},
    CALC(AP_GOR),
    { {"Price",DOUBLE,{100},FORBID},
      {"Delta",DOUBLE,{100},FORBID},
      {" ",PREMIA_NULLTYPE,{0},FORBID}},
    CHK_OPT(AP_GOR),
    CHK_split,
    MET(Init)
};

```

## References