

[Help](#)

```
extern "C" {
#include "
href../../../../mod/nonpar1d/nonpar1d_vol/nonpar1d_vol_h_src.pdfnonpar1d_vol.h"
}

#include <stdio.h>
#include "pnl/pnl_vector.h"
#include <pnl/pnl_finance.h>
#include <
href../../../../common/math/highdim_solver/highdim_vector_h_src.pdfvector>

extern "C" {

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (20167+2) //The "#el
    static int CHK_OPT(AP_NONPAR_LEVY_VOLATILITYSWAP)(void *Opt, void *Mod)
    {
        return NONACTIVE;
    }
    int CALC(AP_NONPAR_LEVY_VOLATILITYSWAP)(void *Opt, void *Mod, PricingMethod *M
    {
        return AVAILABLE_IN_FULL_PREMIA;
    }
#else

typedef struct option {
    double strike;
    double call_price;
    double put_price;
double ivol;
} option;

//compute volatility index in non-parametric Levy model, fn - file name with opt
//option strikes in file must be sorted
//s - asset price
//T - time to maturity
    int levy_vol_product(const char* fn, double s,double Strike, double T, double
{
    const int max_options=50;
    option options[max_options];
```

```

//Reading option data from file fn

FILE* f=fopen(fn, "r");
if (f == NULL)
{
    perror("Could not open file");
    return 0;
}
int n;
fscanf(f, "%d \n", &n);
n=0;
int iatm=0; // index of atm put

/////find k0 at-the-money strike price //////////////////////////////////
while (!feof(f))
{
    fscanf(f, "%lf %lf \n",&options[n].strike,&options[n].ivol);
options[n].call_price=pnl_bs_call(s, options[n].strike, T, r, 0.0, options[n].ivol);
options[n].put_price=pnl_bs_put(s, options[n].strike, T, r, 0.0, options[n].ivol);
    if (n>=1)
        if ((options[n-1].strike<=s) && (s<options[n].strike))
            iatm=n-1;
    n++;
}
double k0=options[iatm].strike; /////the at-the-money strike price
double f_price=k0+exp(r*T)*(options[iatm].call_price-options[iatm].put_price);

fclose(f);

double price_call[max_options],price_put[max_options],k_call[max_options],k_put[max_options];
int n_call=0;
int n_put=0;
for (int i=0;i<=iatm;i++)
    if (options[i].put_price!=0)
    {
        price_put[n_put]=options[i].put_price;
        k_put[n_put]=options[i].strike;
        n_put++;
    }

```

```

    }
    for (int i=iatm+1;i<n;i++)
        if (options[i].call_price!=0)
        {
            price_call[n_call]=options[i].call_price;
            k_call[n_call]=options[i].strike;
            n_call++;
        }
    //////////compute E[ln S_T] //////////
    double log_k0=log(k0);
    double sum1=price_call[0]/pow(k_call[0],2)*(k_call[0]-k_put[n_put-1]);
    for (int i=1;i<n_call;i++)
        sum1+=price_call[i]/pow(k_call[i],2)*(k_call[i]-k_call[i-1]);
    double sum2=price_put[0]/pow(k_put[0],2)*(k_put[1]-k_put[0]);
    for (int i=1;i<n_put;i++)
        sum2+=price_put[i]/pow(k_put[i],2)*(k_put[i]-k_put[i-1]);
    double expected=log_k0+f_price/k0-1-exp(r*T)*(sum1+sum2);
    //////////compute E[ln^2 S_T] //////////
    sum1=price_call[0]*(1-log(k_call[0]))/pow(k_call[0],2)*(k_call[0]-k_put[n_put-1]);
    for (int i=1;i<n_call;i++)
        sum1+=price_call[i]*(1-log(k_call[i]))/pow(k_call[i],2)*(k_call[i]-k_call[i-1]);
    sum2=price_put[0]*(1-log(k_put[0]))/pow(k_put[0],2)*(k_put[1]-k_put[0]);
    for (int i=1;i<n_put;i++)
        sum2+=price_put[i]*(1-log(k_put[i]))/pow(k_put[i],2)*(k_put[i]-k_put[i-1]);
    double expected_quad=pow(log_k0,2)+2*log_k0*(f_price/k0-1)+2*exp(r*T)*(sum1+sum2);
    double var=expected_quad-pow(expected,2);

    *fairval=sqrt(var/T)*100.0;
    *ptprice=*fairval-Strike*exp(-r*T);

return OK;
}

int CALC(AP_NONPAR_LEVY_VOLATILITYSWAP)(void *Opt, void *Mod, PricingMethod *M)
{
    TYPEOPT *ptOpt = (TYPEOPT *)Opt;
    TYPEMOD *ptMod = (TYPEMOD *)Mod;
    double r, strike, spot;
    NumFunc_1 *p;

```

```

r = log(1. + ptMod->R.Val.V_DOUBLE / 100.);
p = ptOpt->PayOff.Val.V_NUMFUNC_1;
strike = p->Par[0].Val.V_DOUBLE;
spot = ptMod->S0.Val.V_DOUBLE;

return levy_vol_product(
    ptMod->implied_volatility.Val.V_FILENAME, spot, strike,
    ptOpt->Maturity.Val.V_DATE, r,
    &(Met->Res[0].Val.V_DOUBLE)/*FAIR STRIKE*/,
    &(Met->Res[1].Val.V_DOUBLE)/*PRICE*/);
}

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

    return WRONG;
}

#endif //PremiaCurrentVersion

static int MET(Init)(PricingMethod *Met, Option *Opt)
{
    Met->HelpFilenameHint = "ap_nonparam_volatilityswap";
    return OK;
}

PricingMethod MET(AP_NONPAR_LEVY_VOLATILITYSWAP) =
{
    "AP_NONPAR_LEVY_VOLATILITYSWAP",
    {{ " ", PREMIA_NULLTYPE, {0}, FORBID}},
    CALC(AP_NONPAR_LEVY_VOLATILITYSWAP),
    { {"Fair strike, in annual volatility points", DOUBLE, {100}, FORBID},
      {"Price", DOUBLE, {100}, FORBID},
      { " ", PREMIA_NULLTYPE, {0}, FORBID}
    },
    CHK_OPT(AP_NONPAR_LEVY_VOLATILITYSWAP),

```

```
        CHK_ok ,  
        MET(Init)  
    } ;  
  
    /*////////////////////////////////////////*/  
}
```