```
   Help
extern "C"{
#include "nonpar1d_vol.h"
}

#include "math/numerics.h"

#include <stdio.h>
#include "pnl/pnl_vector.h"

extern "C"{

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2008+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_NONPAR_VOLATILITYSWAP)(void *Opt, voi
    d *Mod)
{
  return NONACTIVE;
}
int CALC(AP_NONPAR_VOLATILITYSWAP)(void *Opt,void *Mod,
    PricingMethod *Met)
{
return AVAILABLE_IN_FULL_PREMIA;
}
#else


static int readvol(PnlVect **pstrikes, PnlVect **pivol,
    int *nn, char *finname);
//extern double bessi1(double arg);


double bessi0(double x)
{
  double ax,ans;
  double y;  //Accumulate polynomials in double precision

  if ((ax=fabs(x)) < 3.75){ //Polynomial fit
    y = x/3.75;
    y *= y;
```

```
    ans = ax*(1.0 + y*(3.5156229 + y*(3.0899424 + y*(1.2067
    492 + y*(0.2659732
          + y*(0.360768e-1 + y*0.45813e-2))))));
  } else{
    y = 3.75/ax;
    ans = -0.2057706e-1 + y*(0.2635537e-1 + y*(-0.1647633e-
    1 + y*0.392377e-2));
    ans = 0.39894228 + y*(0.1328592e-1 + y*(0.225319e-2+y*(
    -0.157565e-2
                  + y*(0.916281e-2 + y*ans))));
    ans *= (exp(ax)/sqrt(ax));
  }
  return ans;
}
 /*////////////////////////////////////////////////////*/
 int ap_nonpar_volswap(char *ivfname, double S0, double
   Strike, double T, double r, double *fairval, double *ptprice)
 {
 // S0  is a forward price!!
 // Arrays :
 // replStrikes are percentages of forward price, read
   from file
 PnlVect *replStrikes;
 //replOptions are BS vanillas prices for given implied volatility
   PnlVect *replOptions;
 //implVolatil are implied volatilities, read from file
   PnlVect *implVolatil;
 // replWeights are weights of each vanilla option in rep
   licating portfolio
   PnlVect *replWeights;
 // CallPuts are logical indicators to identify the type
   of the option
   PnlVect *CallPuts;

 int flag;
 double kfirst;
 double pvfactor;
 double divid;
 int err_code;

 int k, k0, res, replN;
```

```
double optprice, optdelta, tstrike, tprice;
double wfactor;
double arg, iv0, call0, put0;

// get implied volatility and strikes
replN=0;
divid=0.0;
pvfactor=exp(-r*T);
err_code = readvol(&replStrikes, &implVolatil, &replN,
  ivfname);
if(err_code) return err_code;

replOptions= pnl_vect_create(replN);
replWeights= pnl_vect_create(replN);
CallPuts= pnl_vect_create(replN);

tprice=0.0;
tstrike=S0;
k=0;
flag=1;
//find a separator between call and puts
while((k<replN)&&(flag))
{
  flag=(S0>GET(replStrikes,k));
  LET(CallPuts,k)=!flag;
  k++;
}

k0=k-2;
for(;k<replN;k++)
{
  LET(CallPuts,k)=1;
}
//weights and prices for puts
tstrike=GET(replStrikes,k0+1);
wfactor=0.5*sqrt(0.5*M_PI/S0);

for(k=k0;k>=0;k--)
{
  arg= log( sqrt( GET(replStrikes,k)/S0 ) );
  LET(replWeights,k) =-(GET(replStrikes,k)-tstrike)*(
```

```
  bessi0(arg) - bessi1(arg) );
  LET(replWeights,k) *= wfactor/GET(replStrikes,k)/sq
  rt(GET(replStrikes,k));
  res=pnl_cf_put_bs(S0*pvfactor,GET(replStrikes,k),T,r,
  divid, GET(implVolatil,k)/100.0,  &optprice, &optdelta);

  if(res) {return 1;}
  LET(replOptions,k)=optprice;
  tstrike = GET(replStrikes,k);
  tprice += GET(replOptions,k)*GET(replWeights,k);
}

//weights and prices for calls
tstrike=GET(replStrikes,k0);
for(k=k0+1;k<replN;k++)
{
  arg= log( sqrt( GET(replStrikes,k)/S0 ) );
  LET(replWeights,k) = (GET(replStrikes,k)-tstrike)*(
  bessi1(arg) - bessi0(arg) );
  LET(replWeights,k) *= wfactor/GET(replStrikes,k)/sq
  rt(GET(replStrikes,k));
  res=pnl_cf_call_bs(S0*pvfactor, GET(replStrikes,k),
  T,r,divid, GET(implVolatil,k)/100.0,  &optprice, &optdelta)
  ;

  if(res) {return 1;}
  LET(replOptions,k)=optprice;
  tstrike = GET(replStrikes,k);
  tprice+= GET(replOptions,k)*GET(replWeights,k);
}

//straddle
iv0=GET(implVolatil,k0)+(GET(implVolatil,k0+1)-GET(implV
  olatil,k0))/ (GET(replStrikes,k0+1)-GET(replStrikes,k0))*(
  S0-GET(replStrikes,k0));
res=pnl_cf_call_bs(S0*pvfactor,S0,T,r,divid, iv0/100.0,
   &optprice, &optdelta);

  if(res) {return 1;}
  call0=optprice;
res=pnl_cf_call_bs(S0*pvfactor,S0,T,r,divid, iv0/100.0,
```

```
     &optprice, &optdelta);

    if(res) {return 1;}
    put0=optprice;

  tprice += sqrt(0.5*M_PI)/S0 * ( put0+call0 );

  //portfolio value
  tprice*=100.0/sqrt(T);//*252.0/251.0;

  //fair strike of variance swap, in annual volatility po
    ints
  *fairval= tprice/pvfactor;
  // strike in variance points
  kfirst = pvfactor*Strike;
  // price of var swap
  *ptprice= tprice-kfirst;

  pnl_vect_free(&replStrikes);
  pnl_vect_free(&replOptions);
  pnl_vect_free(&implVolatil);
  pnl_vect_free(&replWeights);
  pnl_vect_free(&CallPuts);

    return OK;
  }

//-------------------------------------------------------
    ----------------
 static int readvol(PnlVect **pstrikes, PnlVect **pivol,
    int *nn, char *finname)
  {
  FILE *filein;
  //std::ifstream fin(finname);

  int i, Nr;

  if((filein=fopen(finname,"r"))==NULL)
  {  printf("Cannot open file %s{n", finname);
    return 0;
  }
```

```
fscanf(filein, "%d ",&Nr);

*nn=Nr;

*pstrikes = pnl_vect_create(Nr);
*pivol = pnl_vect_create(Nr);

for(i=0;i<*nn;i++)
{
  fscanf(filein, "%le %le ",pnl_vect_lget(*pstrikes,i),
   pnl_vect_lget(*pivol, i));
}
fclose(filein);
return 0;
}



int CALC(AP_NONPAR_VOLATILITYSWAP)(void *Opt,void *Mod,
  PricingMethod *Met)
{
  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 ap_nonpar_volswap(
    ptMod->implied_volatility.Val.V_FILENAME, spot, stri
  ke,
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_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_VOLATILITYSWAP)=
  {
    "AP_NONPAR_VOLATILITYSWAP",
    {{" ",PREMIA_NULLTYPE,{0},FORBID}},
    CALC(AP_NONPAR_VOLATILITYSWAP),
    {   {"Fair strike, in annual volatility points",DOUBLE,
    {100},FORBID},
        {"Price",DOUBLE,{100},FORBID},
        {" ",PREMIA_NULLTYPE,{0},FORBID}},
    CHK_OPT(AP_NONPAR_VOLATILITYSWAP),
    CHK_ok ,
    MET(Init)
  } ;

  /*////////////////////////////////////*/
}
```

References