Help

```
#include "optype.h"
#include "pnl/pnl_mathtools.h"
#include "pnl/pnl_cdf.h"
#include "numfunc.h"
/*F(double)*/
double Call(VAR *param,double spot)
{
  double strike=(*param).Val.V_PDOUBLE;

  return MAX(0.,spot-strike);
}

double Put(VAR *param,double spot)
{
  double strike=(*param).Val.V_PDOUBLE;

  return MAX(0.,strike-spot);
}

double CallSpread(VAR *param,double spot)
{
  double strike1=(*param).Val.V_PDOUBLE,strike2=(*(param+1)
    ).Val.V_PDOUBLE;

  return MAX(0.,spot-strike1)-MAX(0.,spot-strike2);
}

double Digit(VAR *param,double spot)
{
  double strike=(*param).Val.V_PDOUBLE,rebate=(*(param+1)).
    Val.V_PDOUBLE;

  return ( (spot>=strike) ? rebate : 0.0);
}

double Zero(VAR *param,double spot)
{
  return 0.;
}
```

```
double Const(VAR *param,double spot)
{
  return param[0].Val.V_DOUBLE;
}

double ConstLim(VAR *param,double spot)
{
  return param[3].Val.V_DOUBLE;
}

double DigitSpecialPayoff(VAR *param,double spot)
{
  double sigmasqrth,d1,critic;
  double r,sigma;
  double k=param[0].Val.V_DOUBLE,h=param[1].Val.V_DOUBLE;

  r=log(1.1);
  sigma=0.2;

  sigmasqrth=sigma*sqrt(h);
  d1=(log(spot/k)+r*h)/sigmasqrth+sigmasqrth/2.;
  critic=k*exp(r*h+sigmasqrth*sigmasqrth/2.);
  if (spot<critic)
    {
      return spot*cdf_nor(d1);
    }
  else
    {
      return spot*cdf_nor(sqrt((4.0*r/(sigma*sigma)+2.)*log
    (spot/k)));
    }

}

/*F(double,double)*/
double BestOf(VAR *param,double spot1,double spot2)
{
  double strike1=(*param).Val.V_PDOUBLE,strike2=(*(param+1)
    ).Val.V_PDOUBLE;

  return MAX(0,MAX(spot1-strike1,spot2-strike2));
```

```
}

double CallMax(VAR *param,double spot1,double spot2)
{
  double strike=(*param).Val.V_PDOUBLE;

  return MAX(0,MAX(spot1,spot2)-strike);
}
double Geom(VAR *param,double spot1,double spot2)
{
  double strike=(*param).Val.V_PDOUBLE;

  return MAX(0,sqrt(spot1*spot2)-strike);
}
double Arim(VAR *param,double spot1,double spot2)
{
  double strike=(*param).Val.V_PDOUBLE;

  return MAX(0,0.5*(spot1+spot2)-strike);
}

double PutMin(VAR *param,double spot1,double spot2)
{
  double strike=(*param).Val.V_PDOUBLE;

  return MAX(0,strike-MIN(spot1,spot2));
}

double Exchange(VAR *param,double spot1,double spot2)
{
  double ratio=(*param).Val.V_PDOUBLE;

  return MAX(0,spot1-ratio*spot2);
}


double Zero2d(VAR *param,double spot1,double spot2)
{
  return 0.;
}
```

```
double Const2d(VAR *param,double spot1,double spot2)
{
  return param[0].Val.V_DOUBLE;
}

double Call_2arg(VAR *param,double spot1,double spot2)
{
  double strike=(*param).Val.V_PDOUBLE;

  return MAX(0,spot1-strike);
}

double Put_2arg(VAR *param,double spot1,double spot2)
{
  double strike=(*param).Val.V_PDOUBLE;

  return MAX(0,strike-spot1);
}

double Call_OverSpot2(VAR *param,double spot1,double spot2)
{
  double strike=(*param).Val.V_PDOUBLE;

  return MAX(0,spot2-strike);
}
double Put_OverSpot2(VAR *param,double spot1,double spot2)
{
  double strike=(*param).Val.V_PDOUBLE;

  return MAX(0,strike-spot2);
}

double Call_StrikeSpot2(VAR *param,double spot1,double spo
    t2)
{
  return MAX(0,spot1-spot2);
}

double Put_StrikeSpot2(VAR *param,double spot1,double spot2
    )
```

```
{
  return MAX(0,spot2-spot1);
}

/*F(double,double,double)*/
double Minimum(VAR *param,double spot,double time)
{
  double minimum=(*(param+4)).Val.V_PDOUBLE;

  return MIN(spot,minimum);
}

double Maximum(VAR *param,double spot,double time)
{
  double maximum=(*(param+4)).Val.V_PDOUBLE;

  return MAX(spot,maximum);
}

double Asian(VAR *param,double spot,double time)
{
  double average=(*(param+4)).Val.V_PDOUBLE,
  frequency=(*(param+2)).Val.V_PDOUBLE,
  starting_date=(*param).Val.V_DOUBLE;

  return ((time-frequency-starting_date)*average+spot*frequ
    ency)/time;
}


double PutBasket_nd(VAR *param,PnlVect *VStock)
{
  double aux=0;
  int i;
  int BS_Dimension=VStock->size;
  double *Stock = VStock->array;
  double Strike = (*param).Val.V_PDOUBLE;
  for (i=0;i<BS_Dimension;i++)
    aux+=Stock[i];
  return MAX(Strike-aux/BS_Dimension,0.);
}
```

```
double CallBasket_nd(VAR *param,PnlVect *VStock)
{
  double aux=0;
  int i;
  int BS_Dimension=VStock->size;
  double *Stock = VStock->array;
  double Strike = (*param).Val.V_PDOUBLE;
  for (i=0;i<BS_Dimension;i++)
    aux+=Stock[i];
  return MAX(aux/BS_Dimension-Strike,0.);
}

double CallMax_nd(VAR *param,PnlVect *VStock)
{
  int i;
  int BS_Dimension=VStock->size;
  double *Stock = VStock->array;
  double Strike = (*param).Val.V_PDOUBLE;
  double aux=Stock[0];
  for (i=1;i<BS_Dimension;i++)
    aux=MAX(aux,Stock[i]);
  return MAX(aux-Strike,0.);
}

double PutMin_nd(VAR *param,PnlVect *VStock)
{
  int i;
  int BS_Dimension=VStock->size;
  double *Stock = VStock->array;
  double Strike = (*param).Val.V_PDOUBLE;
  double aux=Stock[0];
  for (i=1;i<BS_Dimension;i++)
    aux=MIN(aux,Stock[i]);
  return MAX(Strike-aux,0.);
}

double PutGeom_nd(VAR *param,PnlVect *VStock)
{
  int i;
  int BS_Dimension=VStock->size;
```

```
  double *Stock = VStock->array;
  double Strike = (*param).Val.V_PDOUBLE;
  double aux=Stock[0];
  for (i=1;i<BS_Dimension;i++)
    aux*=Stock[i];
  return MAX(Strike-exp(log(aux)/(double)BS_Dimension),0.);
}

double CallGeom_nd(VAR *param,PnlVect *VStock)
{
  int i;
  int BS_Dimension=VStock->size;
  double *Stock = VStock->array;
  double Strike = (*param).Val.V_PDOUBLE;
  double aux=Stock[0];
  for (i=1;i<BS_Dimension;i++)
    aux*=Stock[i];
  return MAX(exp(log(aux)/BS_Dimension)-Strike,0.);
}
```

References