Help

```
#ifndef MATHSB_H
#define MATHSB_H

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "pnl/pnl_mathtools.h"

////////////////////////////////////////////////////////////
    /////////////////
//
// represents a function from R to R by its values in the
    points
// xleft + j*xstep for j=0,...,xnumber-1;
// f(xleft + j*xstep) corresponds to f.val[j]
//
////////////////////////////////////////////////////////////
    //////////////////
typedef struct discrete_fct
{
  double xleft;
  double xstep;
  int xnumber;
  double* val;
} discrete_fct ;

////////////////////////////////////////////////////////////
    /////////////////
//
// represents a function from R to R by its values in the
    points
// xleft + j*xstep for j=0,...,xnumber-1;
// f(xleft + j*xstep) corresponds to f.val[j]
//

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2007+2) //The "#else" part of the code will be freely av
    ailable after the (year of creation of this file + 2)
#else
```

```
double Normal (double mean, double var, double f(double),
    double intervallength, int stepnumber);
// computes E(f(X)), where X is normally distributed N(mea
    n,var)

double NormalTab (double mean, double var, discrete_fct *f)
    ;
// computes E(f(X)), where X is normally distributed N(mea
    n,var)

void Set_discrete_fct (discrete_fct *f, double xleft,
    double xstep, int xnumber);

void SetNf (discrete_fct *g, double var, discrete_fct *f);
// Sets g = NormalTab( ř, var, f) in a reasonable way

//void SetU (discrete_fct *f, double t, double s, discrete_
    fct *g, double xstep);
// Sets f=U_{t,s}g in a reasonable way

double NfUpBound (discrete_fct *f, double var, double vmax)
    ;
// returns the minimum of all x>=f.xleft such that NormalTa
    b(0,var,f*1_{(x,infty)}) < vmax

double NfLoBound (discrete_fct *f, double var, double vmin)
    ;
// returns the minimum of all x<=f.xleft+(f.xnumber-2)*f.x
    step
// such that NormalTab(0,var,f*1_{(x,infty)}) > vmin

double InterpolDiscreteFct(discrete_fct *f, double x);
// returns f(x) via LINEAR interpolation

void ShowDiscreteFct(discrete_fct *f);

void ShowDiscreteFctVal(discrete_fct *f);

void SaveDiscreteFctToFile( discrete_fct *f, char *name);
```

```
void SaveArrayToFile( double *tab, int n, char *name);

void Delete_discrete_fct (discrete_fct *f);

#endif //PremiaCurrentVersion

#endif
```

References