

[Help](#)

```
#ifndef MATHSB_H
#define MATHSB_H

#include <stdio.h>
#include <stdlib.h>
#include <
href../../common/math/cdo/cdo_math_h_src.pdfmath.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 "#els
#else

double Normal(double mean, double var, double f(double), double intervallength,
// computes E(f(X)), where X is normally distributed N(mean,var)
```

```

double NormalTab(double mean, double var, discrete_fct *f);
// computes  $E(f(X))$ , where  $X$  is normally distributed  $N(\text{mean}, \text{var})$ 

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

void SetNf(discrete_fct *g, double var, discrete_fct *f);
// Sets  $g = \text{NormalTab}(0, \text{var}, f)$  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  $x \geq f.xleft$  such that  $\text{NormalTab}(0, \text{var}, f * 1_{\{x, \infty\}})$ 

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

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

```