

## Help

```

#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

#ifdef MATHSB_H
#define MATHSB_H

#include <cmath>
#include <valarray>
#include <iostream>

extern "C" {
#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]
    //
    //////////////////////////////////
    struct discrete_fct
    {
        double xleft;
        double xstep;
        int xnumber;
        double *val;
    };

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  $N(\text{mean}, \text{var})$ 

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

void SetNf(discrete_fct *g, double var, discrete_fct *f);
// Sets  $g = \text{NormalTab}(\check{r}, \text{var}, f)$  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  $\text{NormalTa}$ 
     $b(0, \text{var}, f * 1_{\{(x, \infty)\}}) < \text{vmax}$ 

double NfLoBound(discrete_fct *f, double var, double vmin);
// returns the minimum of all  $x \leq f.xleft + (f.xnumber - 2) * f.x$ 
    step
// 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);

////////////////////////////////////
//                                     //
// Minimization/Maximization of functions //

```

[illegible]

```

//                                     //
////////////////////////////////////

double ScalarProd(std::valarray<double> &x, std::valarray<
    double> &y);
// returns (x*y).sum()

void VectorProd(std::valarray<double> &x, std::valarray<
    double> &mat);
// Supposes that d:=dim(x)>=sqrt(dim(mat));
// sets mat[i*d+j]:=x[i]*x[j] for i,j=0,...,d-1

std::valarray<double> MatrixVectorProd(std::valarray<
    double> &M,
                                     std::valarray<
    double> &x);
// sets d=x.size() and D=M.size()/d
// M is a matrix with D lines and d columns; M_{i,j} = M[i*
    d+j]
// x is a column vector with d entries
// the result M*x is a vector with D entries

#endif

#endif //PremiaCurrentVersion

```

## References