

## [Help](#)

```
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2007+2) //The "#els
#else

#include <stdlib.h>
#include "
href../../mod/hk1d/hk1d_std/mathsb_h_src.pdfmathsb.h"
#include "
href../../common/optype_h_src.pdfoptype.h"
/* defined in premia_obj.c */
extern char premia_data_dir[MAX_PATH_LEN];
extern char *path_sep;

double SQRT_TWO_PI_INV = 0.39894228040143;

void Delete_discrete_fct(discrete_fct *f)
{
    free(f->val);
}

/*//////////////////////////////////////
//
// computes E(f(X)), where X is normally distributed N(mean,var)
// and f is a function double->double
//
// method: Riemann-type sum
// (the integration is restricted to the interval [mean-l,mean+l]
// and then discretized by stepnumber steps)
//
////////////////////////////////////////*/
double Normal(double mean, double var, double f(double), double intervallength,
{
    int j;
    double result = 0;
    double l = intervallength / 2;
    double h = intervallength / stepnumber;
    double oldfvalue = f(mean - l);
    double newfvalue;

    for (j = 0; j < stepnumber; j++)
```



```

    return SQRT_TWO_PI_INV * f->xstep / sqrt(var) * result;
}

```

```

void Set_discrete_fct(discrete_fct *f, double xleft, double xstep, int xnumber)
{
    f->xleft = xleft;
    f->xstep = xstep;
    f->xnumber = xnumber;
    f->val = malloc(xnumber * sizeof(double));
}

```

```

void SetNf(discrete_fct *g, double var, discrete_fct *f)
/* Sets g = NormalTab(0, var, f) such that its domain is the set [RHS>eps]*/
{
    double xleft = -20., xright, eps = 0.0000001, xstep = f->xstep;
    int i;
    /* int xnumber=1;*/

    while ((NormalTab(xleft, var, f) <= eps) && (xleft <= 20)) xleft += 0.25;
    if (NormalTab(xleft, var, f) <= eps)
    {
        printf("Problem in SetNf !\n");
        exit(1);
    }

    if (xstep < 0.001)
    {
        while (NormalTab(xleft, var, f) > eps) xleft -= 0.025;
        while (NormalTab(xleft, var, f) <= eps) xleft += 0.0025;
    }

    while (NormalTab(xleft, var, f) > eps) xleft -= xstep;
    xleft += xstep;
    /* Now we have xleft = min{x; Normal(x,var,f)>eps} */
}

```

```

xright = xleft;
while (NormalTab(xright, var, f) > eps) xright += 0.25;

if (xstep < 0.001)
{
    while (NormalTab(xright, var, f) <= eps) xright -= 0.025;
    while (NormalTab(xright, var, f) > eps) xright += 0.0025;
}

while (NormalTab(xright, var, f) <= eps) xright -= xstep;
/* Now we have xright = max{x; Normal(x,var,f)>eps} */

Set_discrete_fct(g, xleft, (xright - xleft) / (double)(f->xnumber - 1), f->xnu
for (i = 0; i < g->xnumber; i++) g->val[i] = NormalTab(g->xleft + i * g->xstep

}

double NfUpBound(discrete_fct *f, double var, double vmax)
/*returns the minimum of all x>=f.xleft such that NormalTab(0,var,f*1_{{x,infty)
{
    int i, j = 0;
    discrete_fct g;

    if (vmax < 0)
    {
        printf("Stupid call of NfUpBounds !\ n");
        return 20.;
    }

    Set_discrete_fct(&g, f->xleft, f->xstep, f->xnumber);
    for (i = 0; i < g.xnumber; i++) g.val[i] = f->val[i];
    // Now g is a copy of f !!

    while ((NormalTab(0., var, &g) >= vmax) && (j + 99 < f->xnumber))
    {
        for (i = 0; i < 100; i++) g.val[j + i] = 0.;
        j += 100;
    }
}

```

```

if (NormalTab(0., var, &g) >= vmax)
{
    j -= 100;
    printf("%d Problem in NfUpBounds !\n", j);
}

while ((NormalTab(0., var, &g) < vmax) && (j - 10 >= 0))
{
    j -= 10;
    for (i = 0; i < 10; i++) g.val[j + i] = f->val[j + i];
}

while ((NormalTab(0., var, &g) >= vmax) && (j < f->xnumber))
{
    g.val[j] = 0.;
    j++;
}

Delete_discrete_fct(&g);
return f->xleft + j * f->xstep;
}

```

```

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

    Set_discrete_fct(&g, f->xleft, f->xstep, f->xnumber);
    for (j = 0; j < g.xnumber; j++) g.val[j] = 0;

    g.val[g.xnumber - 1] = f->val[g.xnumber - 1];
    j = g.xnumber - 2;
    while ((NormalTab(0, var, &g) <= vmin) && (j >= 0))
    {

```

```

        g.val[j] = f->val[j];
        j--;
        x -= f->xstep;
    }

    if (NormalTab(0., var, &g) <= vmin) printf("Problem in NfLoBounds !\ n");

    Delete_discrete_fct(&g);
    return x;
}

```

```

void ShowDiscreteFct(discrete_fct *f)
{
    printf("xleft    = %f\ n", f->xleft);
    printf("xstep    = %f\ n", f->xstep);
    printf("xnumber = %d\ n", f->xnumber);
    printf("(xright = %f)\ n\ n", f->xleft + (f->xnumber - 1)*f->xstep);
}

```

```

double InterpolDiscreteFct(discrete_fct *f, double x)
/* returny f(x) via LINEAR interpolation */
{
    double xleft = f->xleft, xstep = f->xstep;
    int i = 1, xnumber = f->xnumber;
    double x_i, x_iminus1, c;

    if (x < f->xleft) return 0.;
    if (x > xleft + (xnumber - 1)*xstep) return 0.;
    if (x == xleft + (xnumber - 1)*xstep) return f->val[xnumber - 1];

    while (xleft + i * xstep <= x) i++;
    /* Now we have  x_{i-1} <= x < x_i  and  0<i<xnumber
       Here we denote x_i = xleft+i*xstep */

    x_iminus1 = xleft + (i - 1) * xstep;
    x_i       = xleft +    i * xstep;
    c         = (x_i - x) / (x_i - x_iminus1);
}

```

```

// Now we have  $x = c \cdot x_{i-1} + (1-c) \cdot x_i$  and  $0 < c \leq 1$ 

return c * f->val[i - 1] + (1 - c) * f->val[i];
}

```

```

void ShowDiscreteFctVal(discrete_fct *f)
{
    int j;

    printf("xleft    = %f\ n", f->xleft);
    printf("xstep    = %f\ n", f->xstep);
    printf("xnumber = %d\ n\ n", f->xnumber);

    for (j = 0; j < f->xnumber; j++)
    {
        printf("val[%d] = %e\ n", j, f->val[j]);
        if ((j > 0) && (j % 100 == 0)) getchar();
    }
}

```

```

void SaveDiscreteFctToFile(discrete_fct *f, char *name)
{
    double x;
    int j;
    FILE *ff;
    char data [MAX_PATH_LEN];

    sprintf(data, "%s%s%s", premia_data_dir, path_sep, name);
    ff = fopen(data, "w");

    for (j = 0; j < f->xnumber; j++)
    {
        x = f->xleft + j * f->xstep;
        fprintf(ff, "%f %f\ n", x, f->val[j]);
    }
}

```

```

    }

    fclose(ff);
}

void SaveArrayToFile(double *tab, int n, char *name)
{
    int j;
    FILE *ff;
    char data[MAX_PATH_LEN];
    sprintf(data, "%s%s%s", premia_data_dir, path_sep, name);
    ff = fopen(data, "w");
    for (j = 0; j < n; j++) fprintf(ff, "%d %f\ n", j, tab[j]);
    fclose(ff);
}

#endif //PremiaCurrentVersion

```