第5章　初歩のBNCpackプログラミング

BNCpack(Basic Numerical Computation PACKage)[12] は，著者がチマチマと書き足していった1CPU用数値計算ライブラリである。，IEEE754倍精度及びGMP/MPFRを用いた多倍長精度をサポートしている。本章ではその機能の一部を体験して頂きたい。

5.1　行列計算

ここではごく簡単な行列・ベクトル積

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

を計算するプログラムを作成し，実行してみる。
まず，IEEE754倍精度でこれを計算するプログラムは次のようになる。

```
 1 : #include <stdio.h>
 2 : #include <stdlib.h>
 3 : #include <math.h>
 4 :
 5 : #include "bnc.h"
 6 :
 7 : int main()
 8 : {
 9 :     DVector x, y;
10 :     DMatrix a;
11 :
12 :     a = init_dmatrix(2, 2);
13 :     x = init_dvector(2);
14 :     y = init_dvector(2);
15 :
16 :     set_dmatrix_ij(a, 0, 0, 1.0);
17 :     set_dmatrix_ij(a, 0, 1, 2.0);
```

```
 18 :     set_dmatrix_ij(a, 1, 0, 3.0);
 19 :     set_dmatrix_ij(a, 1, 1, 4.0);
 20 :
 21 :     set_dvector_i(x, 0, 1.0);
 22 :     set_dvector_i(x, 1, 2.0);
 23 :
 24 :     print_dmatrix(a);
 25 :     print_dvector(x);
 26 :
 27 :     mul_dmatrix_dvec(y, a, x);
 28 :
 29 :     print_dvector(y);
 30 :
 31 :     free_dmatrix(a);
 32 :     free_dvector(x);
 33 :     free_dvector(y);
 34 :
 35 :     return EXIT_SUCCESS;
 36 :}
 37 :
```

これをコンパイルし，実行ファイルbnc1を得るMakefileは

```
  1 :CC=gcc
  2 :DEL=rm
  3 :
  4 :INC=-I/usr/local/include
  5 :LIBDIR=-L/usr/local/lib
  6 :LIB=$(LIBDIR) -lbnc -lmpfr -lgmp -lm
  7 :
  8 :bnc1: bnc1.c
  9 :     $(CC) $(INC) -o bnc1 bnc1.c $(LIB)
 10 :
 11 :clean:
 12 :     -$(DEL) bnc1
```

となり，bnc1が生成されれば

```
% ./bnc1
    0   1.0000000000000000e+00   2.0000000000000000e+00
    1   3.0000000000000000e+00   4.0000000000000000e+00

    0   1.0000000000000000e+00
    1   2.0000000000000000e+00
```

```
    0   5.00000000000000000e+00
    1   1.10000000000000000e+01
%
```

という結果を得る。

次にこれを多倍長化した例を見ることにする。10 進 100 桁の多倍長浮動小数点型を使用する。

```
 1 : #include <stdio.h>
 2 : #include <stdlib.h>
 3 : #include <math.h>
 4 :
 5 : #define USE_GMP
 6 : #define USE_MPFR
 7 : #include "bnc.h"
 8 :
 9 : main()
10 : {
11 :     MPFVector x, y;
12 :     MPFMatrix a;
13 :
14 :     set_bnc_default_prec_decimal(100);
15 :
16 :     a = init_mpfmatrix(2, 2);
17 :     x = init_mpfvector(2);
18 :     y = init_mpfvector(2);
19 :
20 :     set_mpfmatrix_ij_d(a, 0, 0, 1.0);
21 :     set_mpfmatrix_ij_d(a, 0, 1, 2.0);
22 :     set_mpfmatrix_ij_d(a, 1, 0, 3.0);
23 :     set_mpfmatrix_ij_d(a, 1, 1, 4.0);
24 :
25 :     set_mpfvector_i_d(x, 0, 1.0);
26 :     set_mpfvector_i_d(x, 1, 2.0);
27 :
28 :     print_mpfmatrix(a);
29 :     print_mpfvector(x);
30 :
31 :     mul_mpfmatrix_mpfvec(y, a, x);
32 :
33 :     print_mpfvector(y);
34 :
35 :     free_mpfmatrix(a);
36 :     free_mpfvector(x);
37 :     free_mpfvector(y);
```

```
 38 :
 39 :     return EXIT_SUCCESS;
 40 :}
 41 :
```

これをコンパイルするMakefileは

```
  1 :CC=gcc
  2 :DEL=rm
  3 :
  4 :INC=-I/usr/local/include
  5 :LIBDIR=-L/usr/local/lib
  6 :LIB=$(LIBDIR) -lbnc -lmpfr -lgmp -lm
  7 :
  8 :all: bnc1 bnc1-1
  9 :
 10 :bnc1: bnc1.c
 11 :     $(CC) $(INC) -ob nc1 bnc1.c $(LIB)
 12 :
 13 :bnc1-1: bnc1-1.c
 14 :     $(CC) $(INC) -o bnc1-1 bnc1-1.c $(LIB)
 15 :
 16 :clean:
 17 :     -$(DEL) bnc1
 18 :     -$(DEL) bnc1-1
```

となり，実行ファイルbnc1-1を起動すると

```
% ./bnc1-1
------------------------------------------------------------------------
BNC Default Precision    : 333 bits(100.2 decimal digits)
BNC Default Rounding Mode: Round to Nearest
------------------------------------------------------------------------
    0 1.0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000 2.0000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
    1 3.0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000 4.0000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
    0 1.0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000
    1 2.0000000000000000000000000000000000000000000000000000000000000000
```

```
0000000000000000000000000000000000000
    0 5.000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000
    1 1.100000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000e1
```

という結果を得る。

5.2 数値積分

最後に，台形則で定積分を計算するプログラムを実行してみる。台形則とは定積分

$$\int_a^b f(x)dx$$

を，積分区間 $[a, b]$ を n 等分割して，区間幅 $h = (b-a)/n$ となった各小区間 $[x_i, x_{i+1}]$ $(x_i = a + ih, i = 0, 1, ..., n-1)$ ごとに台形の面積を

$$\frac{h}{2}(f(x_i) + f(x_{i+1}))$$

として求め，これをすべて加えたものを

$$\int_a^b f(x)dx \approx h\left\{\frac{1}{2}(f(a) + f(b)) + \sum_{i=1}^{n-1} f(x_i)\right\} \tag{5.1}$$

として定積分の近似値として使用する数値積分法の一種である。

$$\int_0^1 x^2 dx$$

を計算するプログラム bnc2.c と，この多倍長版 bnc2-1.c をコンパイルし，実行してみる。

まず，IEEE754 倍精度を用いたプログラム bnc2.c は次のようになる。[0, 1] 区間を 2, 4, 8, 16 分割し，それぞれ台形則を用いて計算している。

```
 1 : #include <stdio.h>
 2 : #include <stdlib.h>
 3 : #include <math.h>
 4 :
 5 : #include "bnc.h"
 6 :
 7 : double dfunc(double x)
```

```
 8 : {
 9 :     return x * x;
10 : }
11 :
12 : int main()
13 : {
14 :     double ans, start, end;
15 :
16 :     start = 0.0;
17 :     end = 1.0;
18 :
19 :     ans = dtrapezoidal_fs(start, end, dfunc, 2);
20 :     printf("%e\n", ans);
21 :     ans = dtrapezoidal_fs(start, end, dfunc, 4);
22 :     printf("%e\n", ans);
23 :     ans = dtrapezoidal_fs(start, end, dfunc, 8);
24 :     printf("%e\n", ans);
25 :     ans = dtrapezoidal_fs(start, end, dfunc, 16);
26 :     printf("%e\n", ans);
27 :
28 :     return EXIT_SUCCESS;
29 : }
```

これを多倍長化すると次のようになる。

```
 1 : #include <stdio.h>
 2 : #include <stdlib.h>
 3 : #include <math.h>
 4 :
 5 : #define USE_GMP
 6 : #define USE_MPFR
 7 : #include "bnc.h"
 8 :
 9 : void dfunc(mpf_t ret, mpf_t x)
10 : {
11 :     mpf_mul(ret, x, x);
12 : }
13 :
14 : int main()
15 : {
16 :     mpf_t ans, start, end, ret;
17 :
18 :     set_bnc_default_prec_decimal(100);
19 :
20 :     mpf_init(ans);
21 :     mpf_init(start);
22 :     mpf_init(end);
```

```
 23 :     mpf_init(ret);
 24 :
 25 :     mpf_set_d(start, 0.0);
 26 :     mpf_set_d(end, 1.0);
 27 :
 28 :     mpf_trapezoidal_fs(ans, start, end, dfunc, 2);
 29 :     mpf_out_str(stdout, 10, 0, ans); printf("\n");
 30 :     mpf_trapezoidal_fs(ans, start, end, dfunc, 4);
 31 :     mpf_out_str(stdout, 10, 0, ans); printf("\n");
 32 :     mpf_trapezoidal_fs(ans, start, end, dfunc, 8);
 33 :     mpf_out_str(stdout, 10, 0, ans); printf("\n");
 34 :     mpf_trapezoidal_fs(ans, start, end, dfunc, 16);
 35 :     mpf_out_str(stdout, 10, 0, ans); printf("\n");
 36 :
 37 :     mpf_clear(ans);
 38 :     mpf_clear(start);
 39 :     mpf_clear(end);
 40 :     mpf_clear(ret);
 41 :
 42 :     return EXIT_SUCCESS;
 43 : }
```

これらをコンパイルし，実行するとそれぞれ

```
% ./bnc2
3.750000e-01
3.437500e-01
3.359375e-01
3.339844e-01
% ./bnc2-1
--------------------------------------------------------------------------------
BNC Default Precision    : 333 bits(100.2 decimal digits)
BNC Default Rounding Mode: Round to Nearest
--------------------------------------------------------------------------------
3.75000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000e-1
3.43750000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000e-1
3.35937500000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000e-1
3.33984375000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000e-1
```

という結果を得る。

演習問題

1. bnc2.c, bnc2-1.c をコンパイルし，それぞれ bnc2, bnc2-1 という実行ファイルを得る Makefile を作れ。

2. bnc1.c, bnc1-1.c をそれぞれ 10 次元, 100 次元, 1000 次元にし，ベンチマークテストを行え。また，bnc1-1.c は 10 進 100 桁, 1000 桁でベンチマークテストを行え。[ヒント: BNCpack でサポートしている `get_secv` 関数を用いるか, UNIX 標準の time コマンドを用いて時間計測を行う。]