数値解析演習第 1 回レポート

14-13394 増田卓斗

1

float 型を用いた結果は以下のようになった。

$n\backslash x$	1	0.1	0.15625
10^2	100.000000	10.000002	15.625000
10^5	100000.000000	9998.556641	15625.000000
10^8	16777216.000000	2097152.000000	4194304.000000

考察:$x = 1$ と $x = 0.15625$ は $10^2, 10^5$ が正確に計算されているが、10^8 は正確に計算できていない。これは大きな数と小さな数を足すために浮動小数点数の指数部を合わせる時に小さな数の情報が落ちてしまい足し算が正確に行われていないためであると考えられる。$x = 0.1$ については $10^2, 10^5$ の時は少し数値にズレがあり、10^8 は正確に計算できていない。10^8 については上の理由と同様で、$10^2, 10^5$ の時のズレは、0.1 を 2 進数で表すと循環小数になり、実数型ではそれを途中で打ち切って計算しているためだと考えられる。

また double 型を用いた結果は以下のようになった。

$n\backslash x$	1	0.1	0.15625
10^2	100.000000	10.000002	15.625000
10^5	100000.000000	10000.000000	15625.000000
10^8	100000000.000000	9999999.981129	15625000.000000

2

float 型を用いた結果は以下のようになった。

n=1: 1.000000

n=2: 0.999999

n=3: 0.999991

n=4: 1.000054

n=5: 1.000990

n=6: 1.009039

n=7: 1.064767

n=8: 0.250000

n=9: 0.031250

考察:$n = 7$ までの値のズレについては、2 進数で表すと循環小数になることが原因で、$n = 8, 9$ については大きな数と小さな数を足したことが原因だと考えられる。また計算結果の精度は $n = 6$ まで、実行時間は $n = 9$ まで耐えられると思った。

3

2^n を int 型を用いて n を 1 から 256 まで動かした結果、$n = 30$ までは正確に計算できたが、$n = 31$ 以降は-2147483648 となった。これは int 型の最大値が $2^31 - 1$ なので、$n = 31$ 以降はオーバーフローして-2147483648 となっている。

2^{-n} は float 型を用いて n を 1 から 256 まで動かした結果、$n = 149$ までは値が計算されているように見えたが、$n = 150$ 以降は 0.000000 となった。ここで、2^{-149} は 1.401298×10^{-45} であった。これは $n = 149$ までは float 型で表すことのできる数で、$n = 150$ 以降はアンダーフローして 0.000000 となっていると考えられる。

アンケート

1. 家
2. MacOS X
3. 言語:C, 環境:gcc
4. 普通だった。量もちょうどいいくらいだった。

ソースコード

問題 1 の float 型を用いたもの。double 型にするには float をすべて double に書き換えればよい。

```
#include <stdio.h>
#include <math.h>
//問題 1
int calc(float x0, int ind) {
    float x=0;
    int i;
    for (i=0; i<pow(10,ind); i++) {
        x += x0;
    }
    printf("n=10^%d: %f\n",ind , x);
    return 0;
}

int main() {
    float x1 = 1;
    float x2 = 0.1;
    float x3 = 0.15625;

    printf("x=1:\n");
    calc(x1,2);
    calc(x1,5);
    calc(x1,8);
    printf("\nx=0.1:\n");
    calc(x2,2);
    calc(x2,5);
    calc(x2,8);
    printf("\nx=0.15625:\n");
    calc(x3,2);
    calc(x3,5);
    calc(x3,8);

    return 0;
}
```

問題 2 のソースコード

```
#include <stdio.h>
#include <math.h>
//問題２
int calc(int n) {
    float x = 0;
    float x0 = powf(10,-n);
    int i;
    for (i=0; i<pow(10,n); i++) {
        x += x0;
    }
    printf("n=%d: %f\n", n, x);
    return 0;
}

int main() {
    int j;
    for (j=1; j<10; j++){
        calc(j);
    }

    return 0;
}
```

問題 3 のソースコード

```
#include <stdio.h>
#include <math.h>
//問題３
int calc1(int n) {
    int x;
    x = powf(2, n);

    printf("n=%d: %d\n", n, x);
    return 0;
}

int calc2(int n) {
    float y;
    y = powf(2, -n);

    printf("n=%d: %e\n", n, y);
    return 0;
}

int main() {
    int j;

    for (j=1; j<=256; j++){
        calc1(j);
    }
    printf("\n");

    for (j=1; j<=256; j++){
        calc2(j);
    }

    return 0;
}
```