情報工学実験Ⅰ実験報告書

プログラミング３

数値計算

作成日：2022 年 7 月 18 日
学籍番号：19AJ088
氏名：髙橋佑輔

１．実験の目的

今までに高等学校や大学において、積分、微分、微分方程式の解析的な手法（数式の変形で解を求める手法）について学んできているが、自然現象をシミュレーションする際などには、解析的な手法が使用できない場合が多い。それに対しコンピュータによる数値計算では、原理上ある程度の誤差はあるものの汎用的に使用することができる。
本実験では数値計算の原理と手法について学び、その原理を確認しプログラムを作成できるようになることを目的とする。

2. 基本事項

２―１．実数の計算と数値計算における誤差 （１）コンピュータによる実数の計算 コンピュータで実数を扱う場合、浮動小数点数を使用することが多い。浮動小数点数は、 仮数、基数および指数でデータを表すが、現在多く使われている手法では基数を固定しており、以下のように表すことが多い。

符号部： 1 bit
仮数部：符号なし整数
指数部：符号付き整数

浮動小数点数では、表したい数値の絶対値を(仮数部)×($基数$)$^{(指数部)}$とする。例えば、 0.5 であれば基数を 10（10 進数）とすると 5.0×10-1 となる。浮動小数点数にはいくつかの方式（フォーマット）があるが、一般的には IEEE 754 により規定された方式が使用されている。

浮動小数点数をプログラムで使用するには、float 型あるいは double 型を使うことが多い。 それぞれの型の変数のサイズは、float 型は 32bit、double 型は 64bit であり、float 型のほう が使用するメモリ容量は少なくて済むが、近年のパソコンでプログラムを実行する場合は、精度の点からメモリ容量を気にせずに double 型を使うことを推奨する。なお、演算速度については、浮動小数点は専用のハードウェアで処理されることが多く、どちらの型を使っても大差はない。

（２）数値計算における誤差 数値計算を行う上で、原理上どうしても生じてしまう誤差がある。以下にその例を示す。

・丸め誤差 実数を有限の桁数の 2 進数で表すために生じる誤差である。10 進数の無理数や循環小数 をコンピュータで扱うときに、この丸め誤差が生じる。また、10 進数では有限小数であっ ても、コンピュータで２進数として扱うことにより循環小数となって丸め誤差

を生じることもある。

例えば、10 進数の 0.1 は 2 進数で表すと、以下のような循環小数となる。

$(0.1)_{(10)} = (0.0001100110011\ldots)_{(2)}$

つまり、10 進数による演算では誤差がないような数値であっても、場合によってはコンピュータで２進数として演算をすると丸め誤差が生じることがある。

・桁落ち

値のほぼ等しい数値同士を減算する場合、有効数字が失われる可能性がある。この現象を 桁落ちと言い、計算誤差につながる。例えば２次方程式の解の公式において、b の絶対値が ac の値よりも非常に大きい場合は桁落ちが起きる。２次方程式の解の公式を式（１）に示す。

$$x = \frac{-b \pm \sqrt{b^2-4ac}}{2a} \quad (1)$$

桁落ちを回避するには、分子の有利化を行えばよい。具体的には、b ＞ 0 の時の式（２）、（３）のようにする。

$$x_1 = \frac{-b \pm \sqrt{b^2-4ac}}{2a} \quad (2)$$

$$x_2 = \frac{-b \pm \sqrt{b^2-4ac}}{2a} \times \frac{b+\sqrt{b^2-4ac}}{b+\sqrt{b^2-4ac}} = -\frac{2c}{b+\sqrt{b^2-4ac}} \quad (3)$$

・情報落ち

絶対値の大きく異なる数値同士の演算において、絶対値の小さな数値が演算結果に反映されない現象がある。これを情報落ちという。例えば、10^{10}のような大きな値に 10^{-8}のよう な小さい値を繰り返し加えると、プログラムのコーディングの方法によっては情報落ちに より正しい結果が得られない場合がある。具体的には、10^{10} に 10^{-8} を 1000 万回加えると、 正しくは 10^{10}+0.1 となるのに、0.1 が反映されず計算結果が 10^{10}となる。

数値計算を行う場合、上述の計算誤差を十分に考慮する必要がある。これらは一般的なコンピュータで数値を扱う上で避けて通れない点である。また、この他にも数値を扱う変数は、その型によって扱える値の範囲が決まっている。それを超えると、いわゆるオーバフロー

（アンダーフロー）を起こし正しい計算結果が得られなくなる。これらの点に留意し数値計算プログラムを作成しなければならない。

2ʷ2．数値計算の具体例：数値計算による積分（区分求積法）

連続関数の不定積分とは、ある関数が与えられているとき、微分すると与えられた関数に一致するような新たな関数（原始関数）を求める操作のことを意味する。すなわち、微分の逆操作が積分（正確には不定積分）であるが、数値計算で積分を扱う場合はいわゆる定積分を計算する。定積分は面積や体積を求める際などに使用される。

数値計算により定積分を求める際に、区分求積法がよく使用される。例えば実数値の連続関数である f(x)を区間[a, b]で定積分するには、その関数の示す面に微小な長方形を敷き詰め、その面積の合計値を求める。区分求積法の概念を図 1 に示す。なお、各矩形の幅 h を刻み幅という。この手法は、原理的に誤差が生じる。(矩形の幅である h を 0 に出来ないため)

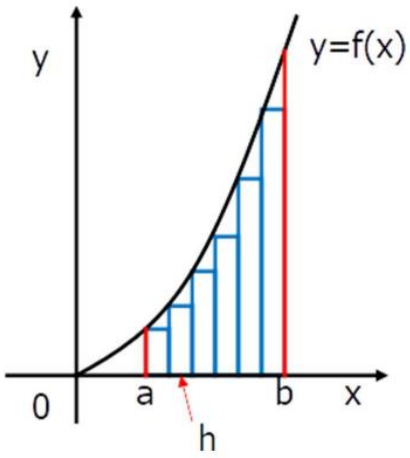

図 1　区分求積法の概念図

以下に、区分求積法のアルゴリズムを説明する。

I. 各変数を初期化する。

刻み幅:h（プログラム上では 0.1 などの具体的数値を代入する）
変数 x=a（定積分の範囲が a〜b のため、初期値は a となる）
定積分値（矩形の面積の合計）：s=0

最初に、図 1 の a から始まる一番左の矩形について面積を求める。

II. y=f(x)について、x を代入し y の値を求める。(初回の計算、すなわち図１の一番左の矩形について計算をするときは、x に a を代入したものが y となる。これは、この矩形の 高さを意味している)

$$y = f(x)$$

III. 矩形の高さ y と刻み幅 h から矩形の面積 ds を求める。

$$ds = y \times h = f(x) \times h$$

IV. 定積分値を格納する変数 s に ds を加え、s の値を更新する。

$$s = s + ds$$

V. x の値に刻み幅 h を加算する。

$$x = x + h$$

VI. 以後、x=b になるまでⅡ～Ⅴの処理を繰り返す。繰返しが完了すると、s には y=f(x)を a～b まで定積分した値（面積）が格納されている。

区分求積法で、関数の示す面に敷き詰める矩形を台形にすると、得られる値の誤差を小さくすることができる。これを台形公式による方法と言う。具体的には f(x)を台形の上底、f(x+h)を台形の下底として計算する。

（以後、検討事項に関する内容）

２－３．数値計算の具体例：微分と常微分方程式の解法（オイラー法）

微分とは、ある関数 f(x)の局所的な変化の情報（変化量）を求めることを意味する。例えば、f(x)が xy 座標平面にグラフとして描画されているなら、この微分は f(x)の接線の傾きを指している。

微分とは微分係数を求めることでもあり、一般的に式（４）のように求められる。ここで、f(x)において上述のように極限が存在する場合、f(x)は a で微分可能といい、この極限を f'(a) と書き x=a における f(x)の微分係数である。その様子を図２に示す。

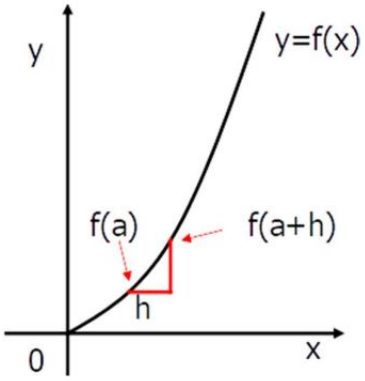

図２　微分の概念図

$$f'(a) = \lim_{x \to a} \frac{f(x)-f(a)}{x-a} = \lim_{h \to 0} \frac{f(a+h)-f(a)}{h}, (x = a + h) \quad (4)$$

数値計算で微分を行うときは、差分を用いる。差分は式（５）のように求められる。式（５）において、h を刻み幅として十分に小さい値をとる必要があるが、0 は使用できない。そのため、誤差が生じる。

$$f'(a) = \frac{f(a+h)-f(a)}{h} \quad (5)$$

現実の世界の物理現象をシミュレートする際、微分方程式を用いることが多い。微分方程式を数値計算により解く場合も、微分を差分として取り扱う。微分方程式を数値計算で解く手法は多数があるがオイラー法について説明する。

・オイラー法による数値計算の実例

１次元の運動シミュレーションとして、地面に向かって降下していく物体の運動をシミュレートする。物体の運動は、式（６）に示す運動方程式で求めることができる。

$$F = ma = m\frac{dv}{dt} = m\frac{d^2x}{dt^2} \quad (6)$$

重力以外の力が働かない自由落下運動の場合、地球上では加速度 α は定数 g=9.80665(m/s^2)となる。自由落下の場合、運動方程式は解析的に容易に求めることができる。速度を v、位置を x とし、速度および位置の初期値をそれぞれ v_0 と x_0 にすれば、以下 の式（７）を求めることができる。

$$v = v_0 + gt$$

$$x = x_0 + v_0 t + \frac{1}{2} g t^2 \qquad (7)$$

このように運動方程式が解析的に解ける場合は、解いた結果の式から値を求めればよい。しかしながら、一般的には運動方程式がいつでも解析的に解けるとは限らない。そこで、式（６）の運動方程式にオイラー法を適用し数値計算により値を求めることにする。 一般に常微分方程式をある初期値のもとで数値計算として解くには、初期値から始めて ある刻み幅で次の値を順々に求めて行く。以下、オイラー法により、式（８）に示す一階常 微分方程式の一般式について解く方法について説明する。

$$\frac{dy}{dx} = f(x, y) \text{ただし、} y(x_0) = y_0 \qquad (8)$$

ここで、刻み幅を h、初期値を x0 および y0 とし、式（９）のように x1 と y1 を近似して求 める。これを順次 x2、y2、x3、y3…として、求めて行けばよい。

$$x_1 = x_0 + h, y_1 = y_0 + f(x_0, y_0)h \qquad (9)$$

自由落下の運動方程式をオイラー法で解く場合は、刻み幅 h は時刻の微小変化を意味し、順次この近似を解いていくことで物体の運動について時間的変化の過程を求めることができる。このとき、二階常微分方程式である式（６）を、式（１０）に示すように v と x に関 する連立一階常微分方程式とする。

$$\frac{dv}{dt} = g$$

$$\frac{dv}{dt} = v \qquad (10)$$

式（１０）に示す通り、自由落下運動では位置 x を時間 t で微分したものが速度 v であ り、その速度 v をさらに時間 t で微分すると重力加速度 g となる（自由落下なので、g 以外 の加速度成分は 0 である)。オイラー法について、具体的なアルゴリズムを以下に示す。

I.　各変数について、初期値を設定する。
刻み幅：h（プログラム上では 0.01 などの具体的数値を代入する）
速度の初期値：v0（プログラム上では 0 などの具体的数値を代入する）
位置の初期値：x0（プログラム上では 100.0 などの具体的数値を代入する）
（重力加速度: g =9.80665 のようにする。これは定数であり変更されない）

II.　オイラー法により、次のステップ（刻み幅 h だけ進んだ時刻）での物体の速度 v1 を求める。

$$v_1 = v_0 + g \cdot h$$

III.　オイラー法により、次のステップでの物体の位置 x1 を求める。（物体は落下するので、 速度の向きをマイナス、すなわち下方向にしている）

$$x_1 = x_0 - v_1 \cdot h$$

IV.　上記の II と III で求めた v1、x1 を用い、同様に v2、x2 を求める。

$$v_2 = v_1 + g \cdot h$$
$$x_2 = x_1 - v_2 \cdot h$$

V.　以下同様に、vi、xi から vi+1、xi+1 を順次求める。

$$v_{i+1} = v_i + g \cdot h$$
$$x_{i+1} = x_i - v_{i+1} \cdot h$$

VI.　適当な終了時刻まで繰り返し演算を行う。

３．実験で使用するプログラミング手法およびクラス

実験では課題を解くプログラムを Java により作成する。本実験の内容は、基本的には四則演算と繰返し処理および計算結果の表示についてプログラムが作成できれば良い。以下に、本実験で使用するおもなクラスを挙げる。詳細は JavaDoc を参照のこと。

・Scanner クラス キーボードから入力された文字を読み取るために使用する。
Scanner オブジェクトの nextDouble()メソッドで実数値、nextInt()メソッドで整数値として読み込む。

（以下、検討事項で使用するクラス）
・FileWriter クラス 新しいファイルを作成したり、既に存在するファイルへテキストデータを追加するため に使用する。

・BufferedWriter クラス

文字をバッファリングすることによって、文字、配列、または文字列を効率良く文字型出 力ストリームに書き込むために使用する。

・PrintWriter クラス オブジェクトの書式付き表現をテキスト出力ストリームに出力するために使用する。

FileWriter クラス、BufferedWriter クラス、PrintWriter クラスを合わせて使用し、数値計算の結果をテキストファイルに出力することができる。

4 - 1．実数値の入出力

実数値の入出力を行うプログラム（EX4_1.java）を以下に示す。これを入力し実行結果を確認しなさい。

```
import java.util.Scanner;

public class Ex4_1{
	public static void main(String[] args){
		System.out.print("実数値を入力してください：");
		Scanner scan = new Scanner(System.in);//キーボード入力の準備
		double num = scan.nextDouble();//キーボードから実数値を入力
		System.out.println("入力した実数値は" + num + "です。");
	}
}
```

4ʷ2．数値計算における丸め誤差

実数値 0.1 を 10 回加算し、その結果を表示するプログラム（Ex4_2.java）を以下に示す。 これを入力し実行結果を確認しなさい。

```
public class Ex4_2{
	public static void main(String[] args){
		int i;
		double sum;
		System.out.println("丸め誤差の確認");
		sum = 0.0;
		for(i=1; i<=10; i++){
			sum = sum + 0.1;
			System.out.println("i:"+ i + " sum:" + sum);
		}
```

```
        }
}
```

４-３．区分求積法による積分

区分求積法を用い、以下の定積分の結果 S を求め表示するプログラムを作成しなさい。刻み幅 h は 0.1 とする。

$$S = \int_0^1 3x^2 dx$$

区分求積法のプログラム（Ex4_3.java）の一部を以下に示す。これを基にプログラムを作成すること。

```
public class Ex4_3{
        public static void main(String[] args){
                double x, y;
                double h;
                double ds ,s;

                h = 0.1;
                x = 0.0;
                s = 0.0;

                for(int i=0; i<(int)(1/h); i++){//h が 0.1 のとき 10 個の矩形を計算する
                        y = 3 * x * x;// y=3x^2 を計算（^2 は 2 乗を意味する）
                        ds = y * h;//一つの矩形領域の面積を計算
                        s = s + ds;//定積分値に矩形面積を加える
                        x = x + h;//x を刻み幅の分だけ増やす
                }
                System.out.println(s);//面積の出力
        }
}
```

５．検討事項

５-１．刻み幅と数値計算誤差の確認

４-３節で作成したプログラム（Ex4_3.java）の刻み幅 h を 0.01 としたプログラム（Ex5_1.java）を作成し、プログラム（Ex4_3.java）との誤差の比較を行いなさい。それぞれの誤差は、理論値と数値計算結果の差から求めることができる。

```
public class Ex5_1{
	public static void main(String[] args){
		int i;
		double x,y;
		double h;
		double ds ,s;
		h = 0.01;
		x = 0.0;
		s = 0.0;
		for(i=0; i<(int)(1/h); i++){
			y = 3 * x * x;
			ds = y * h;
			s = s + ds;
			x = x + h;
		}
		System.out.println(s);
	}
}
```

５―２．台形公式による積分

台形公式を用い、４ｗ３節に示した式の定積分の結果 S を求め表示するプログラム（Ex5_2.java）を作成しなさい。刻み幅 h は 0.1 とする。また、４ｗ３節でのプログラム（Ex4_3.java）と誤差の比較を行いなさい。それぞれの誤差は、理論値と数値計算結果の差 から求めることができる。

```
public class Ex5_2{
	public static void main(String[] args){
		int i;
		double x, y;
		double h;
		double ds, s;
		h = 0.1;
```

```
                x = 0.0;
                s = 0.0;
                for(i=0; i<(int)(1/h); i++){
                        y = 3 * x * x;
                        ds = y * (y + 3*x*h + (h*h)/2);
                        s = s + ds;
                        x = x + h;
                }
                System.out.println(s);
        }
}
```

５－３．運動方程式の解法とシミュレーション（オイラー法）

オイラー法を用いて物体の自由落下のシミュレーションを行うプログラム（Ex5_3.java）を作成しなさい。自由落下のシミュレーションには、２－３で述べたアルゴリズムを使用す ること。このとき時刻の刻み幅 h=0.01、重力加速度 g=9.80665 とし、初速度 v0、初期高度 （位置）x0 は任意の値をプログラム実行時に入力し、その結果を確認できるようにすること。 シミュレーションでは、時刻 t、高度 x および速度 v を求める。時刻 t は、繰り返し処理の 中で t = t + h のように計算すれば求められる。繰返し処理の条件は、x>0、すなわち物体が 地面に接するまでとする。

また、初速度 v0=0(m/s)、初期高度 x0=100(m)としたとき、物体の速さが 20(m/s)に達し た時刻 t(s)を求め、実験結果として記録しなさい。

計算結果である double 型変数について表示桁数を指定する場合（小数点以下何桁にするかなど）は、System.out.printf()や String.format()を利用するとよい。

例 変数 x の値を小数点以下３桁で示す場合（改行あり）。

System.out.printf("%.3f¥n", x);

System.out.println(String.format("%.3f", x));

```
import java.util.Scanner;

public class Ex5_3{
        public static void main(String[] args){
                double t;//時刻
                double x;//高度
                double v;//速度
```

```
                double h;//刻み幅
                double g;//重力加速度

                h = 0.01;
                g = 9.80665;
                t = 0;
                x = 0;
                v = 0;

                System.out.print("初速度を入力してください：");
                Scanner scan1 = new Scanner(System.in);
                double v0 = scan1.nextDouble();
                System.out.print("初期高度を入力してください");
                Scanner scan2 = new Scanner(System.in);
                double   x0 = scan2.nextDouble();

                x = x0;

                while(x > 0){
                        t = t + h;
                        v = v0 + g*t;
                        x = x0 -v*h;

                        System.out.println("時刻:"+ String.format("%.g3",t));
                        System.out.println("高度:"+ String.format("%.3f",x));
                        System.out.println("速度：" + String.format("%.3f",v));
                        x0 = x;
                        v0 = v;
                }
        }
}
```

６．実験結果 作成（入力）したプログラムとともに、それぞれのプログラムの実行結果の数値を見やすいように実験報告書に記載しなさい。

Ex4_1

```
C:¥Users¥81801¥ex4>java Ex4_1
実数値を入力してください：3
入力した実数値は 3.0 です。
```

Ex4_2

```
C:¥Users¥81801¥ex4>java Ex4_2
丸め誤差の確認
i:1 sum:0.1
i:2 sum:0.2
i:3 sum:0.30000000000000004
i:4 sum:0.4
i:5 sum:0.5
i:6 sum:0.6
i:7 sum:0.7
i:8 sum:0.7999999999999999
i:9 sum:0.8999999999999999
i:10 sum:0.9999999999999999
```

Ex4_3

```
C:¥Users¥81801¥ex4>java Ex4_3
0.855
```

Ex5_1

```
C:¥Users¥81801¥ex4>java Ex5_1
0.9850500000000012
```

Ex5_2

```
C:¥Users¥81801¥ex4>java Ex5_2
15.664949999999996
```

Ex5_3

```
C:¥Users¥81801¥ex4>java Ex5_3
初速度を入力してください：1
初期高度を入力してください 2
```

```
Exception in thread "main" java.util.UnknownFormatConversionException: Conversion =
'.'
        at java.util.Formatter.checkText(Formatter.java:2579)
        at java.util.Formatter.parse(Formatter.java:2565)
        at java.util.Formatter.format(Formatter.java:2501)
        at java.util.Formatter.format(Formatter.java:2455)
        at java.lang.String.format(String.java:2940)
        at Ex5_3.main(Ex5_3.java:31)
```

７．実験考察

今回の実験では今までの学生生活で使用してきた公式の原理を理解しその確認を Java プログラミングで出来るようにすることが、目的である。

Scanner クラスを用いることで、キーボードに入力された文字を読み取りを行い数字の値が入るので Ex4_1 では、このクラスの動作が上手くいくかを見てみた。その結果、しっかりと数値が入力できた。ためし、マイナスの値なども打ち込んでみたが問題はなかった。

次に、Ex4_ 2 では 0.1 を 10 回加算していくプログラムで、結果は、概ね大丈夫そうであったが、i:3 sum:0.30000000000000004 の時のこの値は、数値計算上の誤差が発生していると考えられる。その結果、後半の値も

i:8 sum:0.7999999999999999、

i:9 sum:0.8999999999999999

i:10 sum:0.9999999999999999

と少し値が違う結果になってしまっている

Ex4_ 3 の区分求積法を求めるプログラムは、しっかりと動いていると思われる。

８．まとめ

今回の実験を通して感じたのが、自分が過去に習った計算を完全に忘れていたため、プログラムを作ってもいまいち理解をすることができなかった。そこから考えるのは、まずは今回の夏休みの間に微分積分の復習すること。理解をしたらもう一度このプログラムを自分で実装することをしなければいけないと考えた。

今回の実験で理解をしたのが、Java で大多数の計算をすることができるということ、扱うにはそれ相応の理解力が必要と言うことだ。

今回の実験を通して、自分の実力不足を痛感した。

９参考文献

以下は例としての参考文献である．なお，参考した場所に“[1]．”と記述すること．参考文献の書き方は以下の URL の付録 参考文献の記載方法を参考とした．

(https://www.ipsj.or.jp/journal/submit/ronbun_j_prms.html))

[1] 情報処理学会：コンピュータ博物館設立の提言，情報処理学会（オンライン），入手先〈http://www.ipsj.or.jp/03somu/teigen/museum200702.html〉（参照 2007-02-05）.

[2] 阪田史郎（編著）：センサネットワーク，河野隆二：UWB 高速センサネットワーク，pp.79-132，オーム社（2006）.

[3] 荒金陽助，下川清志，金井　敦：音声対話システムにおけるスケーラビリティ評価モデルの検討，情報処理学会論文誌，Vol.46，No.9，pp.2269-2278（2005）.