情報工学実験 II 手順書

ソフトウェア課題
「連続系のシミュレーション (常微分方程式の数値計算)」

version 1.0

愛媛大学工学部情報工学科

2020 年度

第 1 章

はじめに

本実験では常微分方程式の数値計算を C 言語によるプログラムを用いて行う。古くより弾道計算、電気回路の過渡応答解析、近年では宇宙ロケットの軌道計算などなど、常微分方程式の初期値問題に頼られる実際問題は数知れない。そして、それらの問題のほとんどは、解析的に解くことが不可能あるいは困難であり、数値解法に訴えざるをえないものである。

1.1 準備

まず「予習課題」の節で示され課題を実施し、必要な知識を身に付けてから課題に取り組むこと。

1.2 実験

本来の本実験では、第 1 週目で第 2.2 節まで完了し、課題 1・2 の結果を得ることを想定している。同様に第 2 週目では、課題 3・4 の結果を得ることを想定している。予習課題と課題 1・2 を第 1 回目のオンライン授業で完了させる、あるいはその目処をつける必要がある。また、同様に課題 3・4 を第 2 回目のオンライン授業で完了させ、レポートの作成に進めるよう済ませなければならない。

第 2 章

常微分方程式の数値計算

常微分方程式の初期値問題の数値解法というのは、数値積分法の応用であり、まずは、数値積分法を十分理解する必要がある。

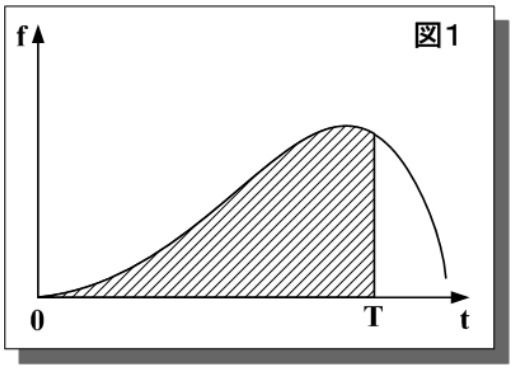
2.1 $dx/dt = ax$ の数値積分法 (一次精度計算)

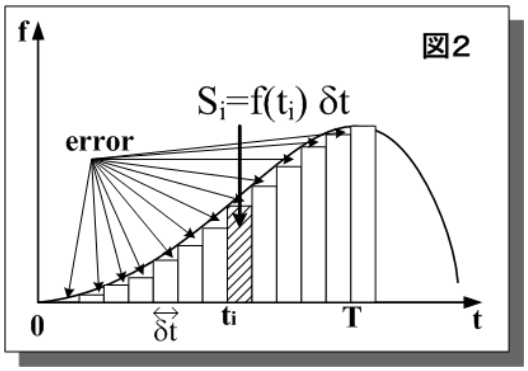
まず、微分方程式

$$\frac{dx}{dt} = f(x) \tag{2.1}$$

を考える。これを解いて x の時間変化 $x(t)$ を求めるには、$x = \int f(x)dt$ の積分を計算すればよい。

コンピューターで積分 (数値積分) するための基本原理は簡単である。
積分 $\int_0^T f(x)dt$ は t 対 $f(x)$ のグラフにおいて、図 1 斜線部の面積を求める事に他ならない。そこで図 2 の様に t 軸方向に分割して小さな四角形を作り、これらの面積の総和を求めればよい。この時、小さく分割するとより正確な面積が求まるが計算量は増え、逆に大きく分割すると粗い近似値しか得られない。





$x(T)$ を求める計算を式で表すと、

$$\begin{aligned} x(T) - x(0) &= \int_{t=0}^{T} f(t)dt \\ &\simeq f(0)\delta t + f(\delta t)\delta t + f(2\delta t)\delta t + \text{.....} + f((n-1)\delta t)\delta t \\ &= \sum_{i=0}^{n-1} f(t_i)\delta t \\ &= x(t_n) - x(t_0) \end{aligned} \tag{2.2}$$

となる。ただし、$T = n\delta t = t_n$ とした。きざみ δt は分割の幅を表している。任意時刻 T の $x(T)$ を知るには n-1 回の足し算が必要になる。

ここで $x(t_1), x(t_2), ..., x(t_n)$ を次々と効率良く求めるには、この式を少し変形した方がよい。具体的には式 (1.2) 中のすべての n を n-1 とした式と式 (1.2) を引き算して、

$$x(t_n) - x(t_{n-1}) = f(t_{n-1})\delta t \tag{2.3}$$

とする。すると $x(t_1)$ は $x(t_1) = x(t_0) + f(t_0)\delta t$ から求まり、$x(t_2)$ は $x(t_2) = x(t_1) + f(t_1)\delta t$ というように $x(t_i)$ は次々求まる。ちなみに式 (1.3) を δt で割ると、δt が十分小さい極限でそれは微分に関する『平均値の定理』を意味している。

以上の内容をプログラムで実現すると”dxdt.c”(4 ページ) のようになる。このプログラムでは、$f(x) = a * x$,$\delta t = dt$ になっている。初めに積分定数として $x(t_0)$ を与えなければならない事に注意せよ (このプログラムでは、適当 ($x0 = 1.0$) に与えてある)。
コンパイルコマンドおよびオプションは、プログラムの始めに書いてある通りである。

コラム（コンパイルオプション）
ここで触れているコンパイルオプション`"-lm"`は、`"libm.a"`ライブラリをコンパイル時にリンクすることを意味します。
この`"libm.a"`というのは数学関数のためのライブラリです。つまり、プログラム中で数学関数を使う場合には、こういったライブラリをリンクしてください。

[予習課題１]

$\frac{dx}{dt} = ax$ の一般解を求めよ。
また、$a = -1$ とし、初期条件として、$t = 0$ において $x = 1$ とした時の解を求めよ。

[課題１]

サンプルプログラム `dxdt.c` をコンパイル・計算実行せよ。
$t = 1.0$ の時点での厳密解 x(1.0) との誤差を dt との関係で評価せよ。
つまり、dt の値を変更し、dt と誤差 ($t = 1$) の関係を調べよ。
(注意、$t = n * dt$ なので、$t = 1.0$ まで計算する時に dt を小さく (大きく) すれば n を大きく (小さく) しなければならない。サンプルプログラムでは、すでに dt から n を計算するようになっている。)
レポートでは、dt と誤差の関係をグラフで表すこと。

```
/*-------------------------------------------------------------*/
/*                    File Name: dxdt.c                        */
/*                compile: gcc dxdt.c -lm                      */
/*-------------------------------------------------------------*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* 数値積分法 1 */

main()
{
   int i, n;
   float t, dt, x0, x1, a;

   /* 初期値設定 */
   t  = 0.0; /* 初期時間 */
   x0 = 1.0; /* 初期位置 */
   a  = -1.0; /* 定数 */

   /* 時間刻みの読み込み */
   printf("INPUT DATA dt ==> ");
   scanf("%f",&dt);

   /* 分割数 n の決定 */
   n=(int)(1.0/dt);

   printf("00\t,t=%f\t,x(t)=%f\n", t, x0);

   for(i=0;i<=n;i++)
   {
    x1 = x0 + a*x0*dt;
    t  = t + dt;

    printf("%02d\t,t=%f\t,x(t)=%f\n", i+1, t, x1);

    x0 = x1;
   }

   printf("THE END\n");
}
```

2.2　$dv/dt = ax$ の数値積分法 (一次精度計算)

次に、微分方程式 $d^2x/dt^2 = f(x)$ を考えよう。
これは力 f(x) のもとでの質点の運動方程式である。変形すると

$$\begin{aligned} dv/dt &= f(x) = ax \\ dx/dt &= v \end{aligned} \tag{2.4}$$

と書ける。2.1 と同様にして、これらそれぞれを同時に解けばよい。

[予習課題２]

$\frac{d^2x}{dt^2} = ax$ の一般解を求めよ。
また、$a = -1$ とし、初期条件として、$t = 0$ において $x = 1, v = 0$ とした時の解を求めよ。

[課題２]

未完成のプログラムを次ページに示す。$f(x) = a * x$ の場合についてサンプルプログラム (`dvdt.c`) をコピーして修正し (****の部分を直し)、完成させよ。また、完成していることを示す結果を提示し、説明しなさい。

```
/*-------------------------------------------------------------*/
/*                     File Name: dvdt.c                       */
/*                  compile: gcc dvdt.c -lm                    */
/*-------------------------------------------------------------*/

#include "stdio.h"
#include "stdlib.h"
#include "math.h"
/* 数値積分法２ */
main()
{
        int   i,n;
        float t,dt,x0,x1,v0,v1,a;

        t = 0.0;
        x0= 1.0;
        v0= 0.0;
        a= -1.0;

        printf("INPUT DATA dt ? \n");
        scanf("%f",&dt);

        n = (int)(1.0/dt);

        printf("00\tt= %f\tx(t)= %f\tv(t)= %f\n",t ,x0 ,v0);

        for(i =0 ; i<=n ; ++i)
        {
                  x1 = ****;
                  v1 = ****;
                  t = t + dt;
                  printf("%02d\tt= %f\tx(t)= %f\tv(t)= %f\n",i+1,t,x1,v1);
                  x0 = x1;
                  v0 = ****;
        }
        printf("THE  END  \n");
}
```

2.3　二天体シミュレーション

二連星や地球と人工衛星の場合など、二天体の運動をシミュレーションしよう。二次元の場合、一般的な運動方程式は、

$$md^2x/dt^2 = F_x, \quad md^2y/dt^2 = F_y \tag{2.5}$$

である。(F_x, F_y) を天体に加わる万有引力とし、m_1, m_2 を天体の質量とする。天体 m_1 については、dvdt.c と同様に、これを

$$m_1 dV_x/dt = F_x, \quad m_1 dV_y/dt = F_y \tag{2.6a}$$

$$dx/dt = V_x, \quad dy/dt = V_y \tag{2.6b}$$

と書き直す。
天体 m_1, m_2 の位置をそれぞれ $(x_1, y_1, z_1), (x_2, y_2, z_2)$ とすると天体 m_1 に働く万有引力は、

$$(F_{x1}, F_{y1}) = (-Gm_1m_2r_x/|r|^3, -Gm_1m_2r_y/|r|^3) \tag{2.7}$$

ただし、$r_x = (x_1 - x_2)$，$r_y = (y_1 - y_2)$，$r = (r_x^2 + r_y^2)^{1/2}$ である。つまり r は 2 天体間の距離である。
一方、m_2 の方は (作用反作用の法則により) 符号を逆にしただけの $(F_{x2}, F_{y2}) = (Gm_1m_2r_x/|r|^3,\ Gm_1m_2r_y/|r|^3)$ とすればよい。

[予習課題３]

C 言語による、外部ファイルへの出力方法について調べよ。

[課題３]

以下の未完成プログラムを完成させ、gnuplot で天体の軌道を作図せよ。また、このプログラムでは、標準出力へ結果を出力している。この結果を、外部ファイルへ出力するように変更しなさい。これまで同様、ファイルは、授業情報のウェブサイトからダウンロードしたものを利用せよ。レポートでは、初期条件等のシミュレーション条件を必ず記述すること。また、この tentai.c の条件では二天体は同一円軌道を描くはずである。同一円軌道にならない場合は、ならない理由を示せ。レポートでは、文章だけでなく、グラフを用いて示すこと。

```
/*-------------------------------------------------------------------*/
/*             File Name: tentai.c                                   */
/*             compile  : gcc tentai.c -lm                           */
/*-------------------------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* 二天体シミュレーション */

/* 万有引力を計算 */
void force(x1, y1, x2, y2, mmg, fx, fy)
float x1,y1,x2,y2,mmg,*fx,*fy;
{
    float    r, xx, yy;

    xx = x1-x2;
    yy = y1-y2;

    r  = sqrt(xx*xx + yy*yy);

    *fx = -mmg*xx/(r*r*r);
    *fy = -mmg*yy/(r*r*r);
}

main()
{
    int    i, j, n;
    float    t, dt, g, fx1, fy1, m1, m2;
    float    x0[2], x1[2], vx0[2], vx1[2];
    float    y0[2], y1[2], vy0[2], vy1[2];
    /* [0]:天体１, [1]:天体２ */

    n  = 200;    /* 時間ステップ数 */
    t  = 0.0;
    dt = 0.05;    /* 時間刻み */
    g  = 4.0;    /* 重力定数 */
    m1 = 1.0; m2 = 1.0;    /* 天体質量 */

    /* 初期位置の設定 */
    x0[0]  =  1.0;  y0[0] = 0.0;
    x0[1]  = -1.0;  y0[1] = 0.0;
    /* 初期速度の設定 */
    vx0[0] =  0.0; vy0[0] =  1.0;
    vx0[1] =  0.0; vy0[1] = -1.0;

    for(i=0;i<n;i++)
    {
        force(x0[0],y0[0],x0[1],y0[1],m1*m2*g,&fx1,&fy1);

        /* dt 後の各天体の位置と速度を求める */
        x1[0] =
        vx1[0]=
        y1[0] =
        vy1[0]=
        x1[1] =
        vx1[1]=
        y1[1] =
        vy1[1]=

        printf("%f\t%f\t%f\t%f\t%f\n", t, x1[0], y1[0], x1[1], y1[1]);

        for(j=0;j<2;j++)
        {
            x0[j]  =
            vx0[j] =
            y0[j]  =
            vy0[j] =
        }

        t  = t + dt;
    }
}
```

[**課題４**]

標準入力等から初期条件 (位置、速度等) を変更できるようにプログラムを修正し、つまり、初期条件を変更するたびにコンパイルしなおす必要がないようにし、さまざまな初期条件により天体の軌道をシミュレーションせよ。
2 つ以上の条件でシミュレーションし、その結果をレポートにすること。また、レポートでは、それぞれの初期条件の設定根拠も同時に示すこと。

[**課題５：発展課題**]

シミュレーション対象の天体の数を柔軟に増やせるように課題 4 で作成したプログラムを改良し、適切なパラメタのもとでシミュレーションを実行した結果を示せ。少なくとも、天体の数が 3、6、10 の場合を調べ、(1) シミュレーション結果がどの程度信頼できる（正しい or 誤差が少ない）と言えるのか、(2) シミュレーション結果からどのようなことが言えるのかを示せ。

[**課題６：発展課題**]

これまで出てきた以外の微分方程式を数値的に解くプログラムを作成せよ。
落下運動や空気抵抗を考慮した落下運動。減衰振動など、どんな問題でも構わない。