計算物理学

3年 0126 三谷 健太 (ミタニ ケンタ)

平成14年8月14日

目 次

第１章　１階の微分方程式を解く

１階の微分方程式 $\Big($ 独立変数 t と未知関数 $x = x\,(t)$ とその導関数 $x'\,(t)$ を含む方程式 $\Big)$ をルンゲ・クッタ $(runge - kutta)$ ４次を使って解く。

$$\frac{d}{dt}x\,(t) = f\Big(t, x\,(t)\Big) \tag{1.1}$$

1.1　１階の微分方程式 - １

関数 f が、 t の関数の場合。

$$\frac{d}{dt}x\,(t) = t = f\,(t) \tag{1.2}$$

1.1.1　解析解を求める。

$$\frac{dx}{dt} = t \tag{1.3}$$

$$\int dx = \int dt \tag{1.4}$$

$$x = \frac{1}{2}t^2 + C_0 \tag{1.5}$$

1.1.2　数値解を求め、解析解と比較する。

初期条件 ： $t = 0$ のとき、 $x\,(0) = 0$ とする。

```
//Ｃ＋＋の標準入出力関数ヘッダファイルをインクルードする。
#include<iostream.h>
//Ｃ＋＋のファイル入出力関数ヘッダファイルをインクルードする。
#include <fstream.h>

//刻み幅 dt を定義する。
#define dt 0.01
//時間 t の初期値を定義する。
#define initial_t 0.0
//位置 x の初期値を定義する。
#define initial_x 0.0
//初期条件より決まる値 c0 を定義する。
#define c0 ( initial_x - initial_t * initial_t / 2.0 )

//ファイル出力のためのファイル名を決め、ファイル出力のためのファイルを開く。
ofstream fileout( "bibunhouteisiki_1.dat" , ios::out );

//関数 f ( 引数 : t , 戻り値 : t )
double f( double t )
{
    return ( t );
}

//関数 runge ( 引数 : t と &x , 戻り値  : なし )
void runge( double t , double &x )
{
    //配列 kx の変数宣言
    double kx[4];

    kx[0] = f( t ) * dt;
    kx[1] = f( t + dt / 2.0 ) * dt;
    kx[2] = f( t + dt / 2.0 ) * dt;
    kx[3] = f( t + dt ) * dt;

    x = x + ( kx[0] + 2.0 * kx[1] + 2.0 * kx[2] + kx[3] ) / 6.0;
}
```

```
//main 関数
int main()
{
    //変数 t と x と ft の変数宣言
    double t, x, ft;

    //時間 t の初期値を代入する。
    t = initial_t;
    //位置 x の初期値を代入する。
    x = initial_x;

    //while 文
    while( t < 10.0 ){
        //解析解を求める。
        ft = t * t / 2.0 + c0;

        //標準出力で解析解と数値解の比較を行なう。
        cout << "\n" << t << "\t" << ft << "\t" << x;
        //ファイル出力で解析解と数値解の比較を行なう。
        fileout << t << "\t" << ft << "\t" << x << "\n";

        //関数 runge の呼び出し。
        runge( t , x );

        t = t + dt;
    }

    cout << "\n";
    //ファイル出力のためのファイルを閉じる。
    fileout.close();

    return ( 0 );
}
```

1.2 １階の微分方程式 - ２

関数 f が、 $x(t)$ の関数の場合。

$$\frac{d}{dt}x(t) = x(t) = f\Big(x(t)\Big) \tag{1.6}$$

1.2.1 解析解を求める。

$$\frac{dx}{dt} = x \tag{1.7}$$

$$\int \frac{1}{x}\,dx = \int dt \tag{1.8}$$

$$\log|x| = t + C_0 \tag{1.9}$$

$$x = \pm e^{t+C_0} \tag{1.10}$$

$$x = Ae^{t} \tag{1.11}$$

1.2.2 数値解を求め、解析解と比較する。

初期条件 ： $t = 0$ のとき、 $x(0) = 1$ とする。

```
//C++の標準入出力関数ヘッダファイルをインクルードする。
#include<iostream.h>
//C++のファイル入出力関数ヘッダファイルをインクルードする。
#include <fstream.h>
//数学関数ヘッダファイルをインクルードする。
#include<math.h>

//刻み幅 dt を定義する。
#define dt 0.01
//時間 t の初期値を定義する。
#define initial_t 0.0
//位置 x の初期値を定義する。
#define initial_x 1.0
//初期条件より決まる値 A を定義する。
#define A ( initial_x / exp( initial_t ) )

//ファイル出力のためのファイル名を決め、ファイル出力のためのファイルを開く。
ofstream fileout( "bibunhouteisiki_2.dat" , ios::out );

//関数 f ( 引数 : x , 戻り値 : x )
double f( double x )
{
```

```
    return ( x );
}

//関数 runge ( 引数 : &x , 戻り値  : なし )
void runge( double &x )
{
    //配列 kx の変数宣言
    double kx[4];

    kx[0] = f( x ) * dt;
    kx[1] = f( x + kx[0] / 2.0 ) * dt;
    kx[2] = f( x + kx[1] / 2.0 ) * dt;
    kx[3] = f( x + kx[2] ) * dt;

    x = x + ( kx[0] + 2.0 * kx[1] + 2.0 * kx[2] + kx[3] ) / 6.0;
}

//main 関数
int main()
{
    //変数 t と x と ft の変数宣言
    double t, x, ft;

    //時間 t の初期値を代入する。
    t = initial_t;
    //位置 x の初期値を代入する。
    x = initial_x;

    //while 文
    while( t < 10.0 ){
        //解析解を求める。
        ft = A * exp( t );

        //標準出力で解析解と数値解の比較を行なう。
        cout << "\n" << t << "\t" << ft << "\t" << x;
        //ファイル出力で解析解と数値解の比較を行なう。
        fileout << t << "\t" << ft << "\t" << x << "\n";

        //関数 runge の呼び出し。
        runge( x );

        t = t + dt;
    }

    cout << "\n";
    //ファイル出力のためのファイルを閉じる。
    fileout.close();

    return ( 0 );
}
```

1.3　１階の微分方程式 - ３

関数 f が、 t と $x\left(t\right)$ の関数の場合。

$$\frac{d}{dt}x\left(t\right)=tx\left(t\right)=f\Big(t,x\left(t\right)\Big) \tag{1.12}$$

1.3.1　解析解を求める。

$$\begin{aligned}
\frac{dx}{dt} &= tx && (1.13)\\
\int\frac{1}{x}\,dx &= \int t\,dt && (1.14)\\
\log\left|x\right| &= \frac{1}{2}t^2+C_0 && (1.15)\\
x &= \pm e^{\frac{1}{2}t^2+C_0} && (1.16)\\
x &= Ae^{\frac{1}{2}t^2} && (1.17)
\end{aligned}$$

1.3.2　数値解を求め、解析解と比較する。

初期条件 ： $t=0$ のとき、 $x\left(0\right)=1$ とする。

```
//C++の標準入出力関数ヘッダファイルをインクルードする。
#include<iostream.h>
//C++のファイル入出力関数ヘッダファイルをインクルードする。
```

```
#include <fstream.h>
//数学関数ヘッダファイルをインクルードする。
#include<math.h>

//刻み幅 dt を定義する。
#define dt 0.01
//時間 t の初期値を定義する。
#define initial_t 0.0
//位置 x の初期値を定義する。
#define initial_x 1.0
//初期条件より決まる値 A を定義する。
#define A ( initial_x / exp( initial_t * initial_t / 2.0 ) )

//ファイル出力のためのファイル名を決め、ファイル出力のためのファイルを開く。
ofstream fileout( "bibunhouteisiki_3.dat" , ios::out );

//関数 f ( 引数 : t と x , 戻り値 : ( t * x ) )
double f( double t , double x )
{
    return ( t * x );
}

//関数 runge ( 引数 : t と &x , 戻り値  : なし )
void runge( double t , double &x )
{
    //配列 kx の変数宣言
    double kx[4];

    kx[0] = f( t , x ) * dt;
    kx[1] = f( t + dt / 2.0 , x + kx[0] / 2.0 ) * dt;
    kx[2] = f( t + dt / 2.0 , x + kx[1] / 2.0 ) * dt;
    kx[3] = f( t + dt , x + kx[2] ) * dt;

    x = x + ( kx[0] + 2.0 * kx[1] + 2.0 * kx[2] + kx[3] ) / 6.0;
}

//main 関数
int main()
{
    //変数 t と x と ft の変数宣言
    double t, x, ft;

    //時間 t の初期値を代入する。
    t = initial_t;
    //位置 x の初期値を代入する。
    x = initial_x;

    //while 文
    while( t < 10.0 ){
        //解析解を求める。
        ft = A * exp( t * t / 2.0 );

        //標準出力で解析解と数値解の比較を行なう。
        cout << "\n" << t << "\t" << ft << "\t" << x;
        //ファイル出力で解析解と数値解の比較を行なう。
        fileout << t << "\t" << ft << "\t" << x << "\n";

        //関数 runge の呼び出し。
        runge( t , x );

        t = t + dt;
    }

    cout << "\n";
    //ファイル出力のためのファイルを閉じる。
    fileout.close();

    return ( 0 );
}
```

1.4　１階の微分方程式 - 4

$x(t) = \sin(t)$ を求める。

$$\frac{d}{dt}x(t) = \cos(t) = f(t) \tag{1.18}$$

1.4.1　解析解を求める。

１階の微分方程式の場合は、任意の値を取り得る定数 ( ) 任意定数 または 積分定数 () を１個含む解のことを一般解という。また、一般解に初期条件を与えて得られる解のことを特殊解という。

一般解を求める。

一般解は、任意定数 C_0 をもつ。

$$\frac{dx}{dt} = \cos(t) \tag{1.19}$$
$$\int dx = \int \cos(t)\ dt \tag{1.20}$$
$$x = \sin(t) + C_0 \tag{1.21}$$
$$x(t) = \sin(t) + C_0 \tag{1.22}$$

特殊解を求める。

特殊解は、一般解 (1.22) に初期条件 ： $t = 0$ のとき、 $x(0) = 0$ を与えて得られる。

$$x(0) = \sin(0) + C_0 \tag{1.23}$$
$$0 = 0 + C_0 \tag{1.24}$$
$$C_0 = 0 \tag{1.25}$$
$$x(t) = \sin(t) \tag{1.26}$$

1.4.2 数値解を求め、解析解と比較する。

初期条件 ： $t = 0$ のとき、 $x(0) = 0$ とする。また、任意定数 ： $c_0 = 0$ とする。

```
//C＋＋の標準入出力関数ヘッダファイルをインクルードする。
#include<iostream.h>
//C＋＋のファイル入出力関数ヘッダファイルをインクルードする。
#include <fstream.h>
//数学関数ヘッダファイルをインクルードする。
#include<math.h>

//刻み幅 dt を定義する。
#define dt M_PI/180
//時間 t の初期値を定義する。
#define initial_t 0.0
//位置 x の初期値を定義する。
#define initial_x 0.0
//任意定数 c0 を定義する。
#define c0 0.0

//ファイル出力のためのファイル名を決め、ファイル出力のためのファイルを開く。
ofstream fileout( "bibunhouteisiki_4.dat" , ios::out );

//関数 f ( 引数 : t , 戻り値 : cos( t ) )
double f( double t )
{
    return ( cos( t ) );
}

//関数 runge ( 引数 : t と &x , 戻り値  : なし )
void runge( double t , double &x )
{
    //配列 kx の変数宣言
    double kx[4];

    kx[0] = f( t ) * dt;
    kx[1] = f( t + dt / 2.0 ) * dt;
    kx[2] = f( t + dt / 2.0 ) * dt;
    kx[3] = f( t + dt ) * dt;

    x = x + ( kx[0] + 2.0 * kx[1] + 2.0 * kx[2] + kx[3] ) / 6.0;
}

//main 関数
int main()
{
    //変数 t と x と ft の変数宣言
    double t, x, ft;

    //時間 t の初期値を代入する。
    t = initial_t;
    //位置 x の初期値を代入する。
    x = initial_x;

    //while 文
    while( t < 2.0 * M_PI ){
        //解析解を求める。
        ft = sin( t ) + c0;

        //標準出力で解析解と数値解の比較を行なう。
```

```
        cout << "\n" << t << "\t" << ft << "\t" << x;
        //ファイル出力で解析解と数値解の比較を行なう。
        fileout << t << "\t" << ft << "\t" << x << "\n";

        //関数 runge の呼び出し。
        runge( t , x );

        t = t + dt;
    }

    cout << "\n";
    //ファイル出力のためのファイルを閉じる。
    fileout.close();

    return ( 0 );
}
```

第2章　２階の微分方程式を解く

物理現象 - 速度に比例する抵抗を受ける落下 の 運動方程式

$$m\frac{d^2}{dt^2}x\left(t\right) = mg - k\frac{d}{dt}x\left(t\right) \tag{2.1}$$

をルンゲ・クッタ $\left(runge-kutta\right)$ ４次を使って解く。ここで、 m は質点の質量、 g は重力加速度、 k は抵抗の定数のことである。

2.1　１階の微分方程式 - １

速度 $vx\left(t\right)$ を求める。

$$\frac{d}{dt}vx\left(t\right) = g - \frac{k}{m}vx\left(t\right) = f\Big(vx\left(t\right)\Big) \tag{2.2}$$

2.1.1　解析解を求める。

$$\begin{aligned}
\frac{dvx}{dt} &= g - \frac{k}{m}vx && (2.3)\\
\frac{dvx}{dt} &= \frac{k}{m}\left(\frac{mg}{k} - vx\right) && (2.4)\\
\int \frac{1}{\frac{mg}{k} - vx}\, dvx &= \frac{k}{m}\int dt && (2.5)\\
\log\left|\frac{mg}{k} - vx\right| &= -\frac{k}{m}t + C_0 && (2.6)\\
\frac{mg}{k} - vx &= \pm e^{-\frac{k}{m}t + C_0} && (2.7)\\
vx &= A_1 e^{-\frac{k}{m}t} + \frac{mg}{k} && (2.8)
\end{aligned}$$

2.1.2　数値解を求め、解析解と比較する。

初期条件 ： $t = 0$ のとき、 $vx\left(0\right) = 0$ とする。

```
//C＋＋の標準入出力関数ヘッダファイルをインクルードする。
#include<iostream.h>
//C＋＋のファイル入出力関数ヘッダファイルをインクルードする。
#include <fstream.h>
//数学関数ヘッダファイルをインクルードする。
#include<math.h>

//質点の質量 m を定義する。
#define m 1.0
//重力加速度 g を定義する。
#define g 9.8
//抵抗の定数 k を定義する。
#define k 1.0
//刻み幅 dt を定義する。
#define dt 0.01
//時間 t の初期値を定義する。
#define initial_t 0.0
//速度 vx の初期値を定義する。
#define initial_vx 0.0
//任意定数 A1 を定義する。
#define A1 ( ( initial_vx - m * g / k ) / exp( k * initial_t / m ) )

//ファイル出力のためのファイル名を決め、ファイル出力のためのファイルを開く。
ofstream fileout( "ikkaino_1.dat" , ios::out );
```

```
//関数 fax : 加速度
double fax( double vx )
{
    return ( g - k * vx / m );
}

//関数 runge ( 引数 : &vx , 戻り値  : なし )
void runge( double &vx )
{
    //配列 kvx の変数宣言
    double kvx[4];

    kvx[0] = fax( vx ) * dt;
    kvx[1] = fax( vx + kvx[0] / 2.0 ) * dt;
    kvx[2] = fax( vx + kvx[1] / 2.0 ) * dt;
    kvx[3] = fax( vx + kvx[2] ) * dt;

    vx = vx + ( kvx[0] + 2.0 * kvx[1] + 2.0 * kvx[2] + kvx[3] ) / 6.0;
}

//main 関数
int main()
{
    //変数 t と vx と ft_vx の変数宣言
    double t, vx, ft_vx;

    //時間 t の初期値を代入する。
    t = initial_t;
    //速度 vx の初期値を代入する。
    vx = initial_vx;

    //while 文
    while( t <= 20.0 ){
        //速度 vx の解析解を求める。
        ft_vx = A1 * exp( - k * t / m ) + m * g / k;

        //標準出力で解析解と数値解の比較を行なう。
        cout << "\n" << t << "\t" << ft_vx << "\t" << vx;
        //ファイル出力で解析解と数値解の比較を行なう。
        fileout << t << "\t" << ft_vx << "\t" << vx << "\n";

        //関数 runge の呼び出し。
        runge( vx );

        t = t + dt;
    }

    cout << "\n";
    //ファイル出力のためのファイルを閉じる。
    fileout.close();

    return ( 0 );
}
```

2.2　１階の微分方程式 - ２

位置 $x\left(t\right)$ を求める。

$$\frac{d}{dt}x\left(t\right)=-\frac{mg}{k}e^{-\frac{k}{m}t}+\frac{mg}{k}=f\left(t\right) \tag{2.9}$$

2.2.1　解析解を求める。

$$\begin{aligned}\frac{dx}{dt} &= -\frac{mg}{k}e^{-\frac{k}{m}t}+\frac{mg}{k} &(2.10)\\ \frac{dx}{dt} &= \frac{mg}{k}\left(1-e^{-\frac{k}{m}t}\right) &(2.11)\\ \int dx &= \frac{mg}{k}\int\left(1-e^{-\frac{k}{m}t}\right)\,dt &(2.12)\\ x &= \frac{m^2g}{k^2}e^{-\frac{k}{m}t}+\frac{mg}{k}t+A_2 &(2.13)\end{aligned}$$

2.2.2　数値解を求め、解析解と比較する。

初期条件 ： $t=0$ のとき、 $x\left(0\right)=0$ とする。

```
//C＋＋の標準入出力関数ヘッダファイルをインクルードする。
#include<iostream.h>
```

```
//C＋＋のファイル入出力関数ヘッダファイルをインクルードする。
#include <fstream.h>
//数学関数ヘッダファイルをインクルードする。
#include<math.h>

//質点の質量 m を定義する。
#define m 1.0
//重力加速度 g を定義する。
#define g 9.8
//抵抗の定数 k を定義する。
#define k 1.0
//刻み幅 dt を定義する。
#define dt 0.01
//時間 t の初期値を定義する。
#define initial_t 0.0
//位置 x の初期値を定義する。
#define initial_x 0.0
//任意定数 A2 を定義する。
#define A2 ( initial_x - m * m * g * exp( - k * initial_t / m ) / ( k * k ) - m * g * initial_t / k )

//ファイル出力のためのファイル名を決め、ファイル出力のためのファイルを開く。
ofstream fileout( "ikkaino_2.dat" , ios::out );

//関数 fvx : 速度
double fvx( double t )
{
    return ( m * g * ( 1 - exp( - k * t / m ) ) / k );
}

//関数 runge ( 引数 : t と &x , 戻り値  : なし )
void runge( double t , double &x )
{
    //配列 kx の変数宣言
    double kx[4];

    kx[0] = fvx( t ) * dt;
    kx[1] = fvx( t + dt / 2.0 ) * dt;
    kx[2] = fvx( t + dt / 2.0 ) * dt;
    kx[3] = fvx( t + dt ) * dt;

    x = x + ( kx[0] + 2.0 * kx[1] + 2.0 * kx[2] + kx[3] ) / 6.0;
}

//main 関数
int main()
{
    //変数 t と x と ft_x の変数宣言
    double t , x , ft_x;

    //時間 t の初期値を代入する。
    t = initial_t;
    //位置 x の初期値を代入する。
    x = initial_x;

    //while 文
    while( t <= 20.0 ){
        //位置 x の解析解を求める。
        ft_x = m * m * g * exp( - k * t / m ) / ( k * k ) + m * g * t / k + A2;

        //標準出力で解析解と数値解の比較を行なう。
        cout << "\n" << t << "\t" << ft_x << "\t" << x;
        //ファイル出力で解析解と数値解の比較を行なう。
        fileout << t << "\t" << ft_x << "\t" << x << "\n";

        //関数 runge の呼び出し。
        runge( t , x );

        t = t + dt;
    }

    cout << "\n";
    //ファイル出力のためのファイルを閉じる。
    fileout.close();

    return ( 0 );
}
```

2.3　２階の微分方程式 - １

$$\begin{cases} \dfrac{d}{dt}vx\left(t\right) = g - \dfrac{k}{m}vx\left(t\right) = f\Big(vx\left(t\right)\Big) \\ \dfrac{d}{dt}x\left(t\right) = A_1 e^{-\frac{k}{m}t} + \dfrac{mg}{k} = f\left(t\right) \end{cases} \tag{2.14}$$

初期条件 ： $t = 0$ のとき、 $vx\left(0\right) = 0$ 、 $x\left(0\right) = 0$ とする。

```
//C＋＋の標準入出力関数ヘッダファイルをインクルードする。
```

```
#include<iostream.h>
//C++のファイル入出力関数ヘッダファイルをインクルードする。
#include <fstream.h>
//数学関数ヘッダファイルをインクルードする。
#include<math.h>

//質点の質量 m を定義する。
#define m 1.0
//重力加速度 g を定義する。
#define g 9.8
//抵抗の定数 k を定義する。
#define k 1.0
//刻み幅 dt を定義する。
#define dt 0.01
//時間 t の初期値を定義する。
#define initial_t 0.0
//速度 vx の初期値を定義する。
#define initial_vx 0.0
//位置 x の初期値を定義する。
#define initial_x 0.0
//任意定数 A1 を定義する。
#define A1 ( ( initial_vx - m * g / k ) / exp( - k * initial_t / m ) )
//任意定数 A2 を定義する。
#define A2 ( initial_x + A1 * m * exp( - k * initial_t / m ) / k - m * g * initial_t / k )

//ファイル出力のためのファイル名を決め、ファイル出力のためのファイルを開く。
ofstream fileout( "nikaino_1.dat" , ios::out );

//関数 fax : 加速度
double fax( double vx )
{
    return ( g - k * vx / m );
}

//関数 runge_ax ( 引数 : &vx , 戻り値  : なし )
void runge_ax( double &vx )
{
    //配列 kvx の変数宣言
    double kvx[4];

    kvx[0] = fax( vx ) * dt;
    kvx[1] = fax( vx + kvx[0] / 2.0 ) * dt;
    kvx[2] = fax( vx + kvx[1] / 2.0 ) * dt;
    kvx[3] = fax( vx + kvx[2] ) * dt;

    vx = vx + ( kvx[0] + 2.0 * kvx[1] + 2.0 * kvx[2] + kvx[3] ) / 6.0;
}

//関数 fvx : 速度
double fvx( double t )
{
    return ( A1 * exp( - k * t / m ) + m * g / k );
}

//関数 runge_vx ( 引数 : t と &x , 戻り値  : なし )
void runge_vx( double t , double &x )
{
    //配列 kx の変数宣言
    double kx[4];

    kx[0] = fvx( t ) * dt;
    kx[1] = fvx( t + dt / 2.0 ) * dt;
    kx[2] = fvx( t + dt / 2.0 ) * dt;
    kx[3] = fvx( t + dt ) * dt;

    x = x + ( kx[0] + 2.0 * kx[1] + 2.0 * kx[2] + kx[3] ) / 6.0;
}

//main 関数
int main()
{
    //変数 t と vx と x と ft_vx と ft_x の変数宣言
    double t, vx, x, ft_vx, ft_x;

    //時間 t の初期値を代入する。
    t = initial_t;
    //速度 vx の初期値を代入する。
    vx = initial_vx;
    //位置 x の初期値を代入する。
    x = initial_x;

    //while 文
    while( t < 20.0 ){
        //速度 vx の解析解を求める。
        ft_vx = A1 * exp( - k * t / m ) + m * g / k;
        //位置 x の解析解を求める。
        ft_x = - A1 * k * exp( - k * t / m ) / m + m * g * t / k + A2;

        //標準出力で解析解と数値解の比較を行なう。
        cout << "\n" << t << "\t" << ft_vx << "\t" << vx << "\t" << ft_x << "\t" << x;
        //ファイル出力で解析解と数値解の比較を行なう。
        fileout << t << "\t" << ft_vx << "\t" << vx << "\t" << ft_x << "\t" << x << "\n";

        //関数 runge_ax の呼び出し。
        runge_ax( vx );
        //関数 runge_vx の呼び出し。
        runge_vx( t , x );

        t = t + dt;
```

```
    }

    cout << "\n";
    //ファイル出力のためのファイルを閉じる。
    fileout.close();

    return ( 0 );
}
```

2.4　２階の微分方程式 - 2

$$\begin{cases} \dfrac{d}{dt}vx\,(t) = g - \dfrac{k}{m}vx\,(t) = f\Big(vx\,(t)\Big) \\ \dfrac{d}{dt}x\,(t) = vx\,(t) = f\Big(vx\,(t)\Big) \end{cases} \tag{2.15}$$

初期条件 ： $t = 0$ のとき、 $vx\,(0) = 0$ 、 $(0) = 0$ とする。

```
//C＋＋の標準入出力関数ヘッダファイルをインクルードする。
#include<iostream.h>
//C＋＋のファイル入出力関数ヘッダファイルをインクルードする。
#include <fstream.h>
//数学関数ヘッダファイルをインクルードする。
#include<math.h>

//質点の質量 m を定義する。
#define m 1.0
//重力加速度 g を定義する。
#define g 9.8
//抵抗の定数 k を定義する。
#define k 1.0
//刻み幅 dt を定義する。
#define dt 0.01
//時間 t の初期値を定義する。
#define initial_t 0.0
//速度 vx の初期値を定義する。
#define initial_vx 0.0
//位置 x の初期値を定義する。
#define initial_x 0.0
//任意定数 A1 を定義する。
#define A1 ( ( initial_vx - m * g / k ) / exp( - k * initial_t / m ) )
//任意定数 A2 を定義する。
#define A2 ( initial_x + A1 * m * exp( - k * initial_t / m ) / k - m * g * initial_t / k )

//ファイル出力のためのファイル名を決め、ファイル出力のためのファイルを開く。
ofstream fileout( "nikaino_2.dat" , ios::out );

//関数 fax : 加速度
double fax( double vx )
{
    return ( g - k * vx / m );
}

//関数 fvx : 速度
double fvx( double vx )
{
    return ( vx );
}

//関数 runge ( 引数 : &vx と &x , 戻り値  : なし )
void runge( double &vx , double &x )
{
    //配列 kvx と kx の変数宣言
    double kvx[4], kx[4];

    kvx[0] = fax( vx ) * dt;
    kx[0] = fvx( vx ) * dt;
    kvx[1] = fax( vx + kvx[0] / 2.0 ) * dt;
    kx[1] = fvx( vx + kvx[0] / 2.0 ) * dt;
    kvx[2] = fax( vx + kvx[1] / 2.0 ) * dt;
    kx[2] = fvx( vx + kvx[1] / 2.0 ) * dt;
    kvx[3] = fax( vx + kvx[2] ) * dt;
    kx[3] = fvx( vx + kvx[2] ) * dt;

    vx = vx + ( kvx[0] + 2.0 * kvx[1] + 2.0 * kvx[2] + kvx[3] ) / 6.0;
    x = x + ( kx[0] + 2.0 * kx[1] + 2.0 * kx[2] + kx[3] ) / 6.0;
}

//main 関数
int main()
{
    //変数 t と vx と x と ft_vx と ft_x の変数宣言
    double t , vx , x , ft_vx , ft_x;

    //時間 t の初期値を代入する。
    t = initial_t;
    //速度 vx の初期値を代入する。
```

```
    vx = initial_vx;
    //位置 x の初期値を代入する。
    x = initial_x;

    //while 文
    while( t < 20.0 ){
        //速度 vx の解析解を求める。
        ft_vx = A1 * exp( - k * t / m ) + m * g / k;
        //位置 x の解析解を求める。
        ft_x = - A1 * k * exp( - k * t / m ) / m + m * g * t / k + A2;

        //標準出力で解析解と数値解の比較を行なう。
        cout << "\n" << t << "\t" << ft_vx << "\t" << vx << "\t" << ft_x << "\t" << x;
        //ファイル出力で解析解と数値解の比較を行なう。
        fileout << t << "\t" << ft_vx << "\t" << vx << "\t" << ft_x << "\t" << x << "\n";

        //関数 runge の呼び出し。
        runge( vx , x );

        t = t + dt;
    }

    cout << "\n";
    //ファイル出力のためのファイルを閉じる。
    fileout.close();

    return ( 0 );
}
```

第3章　１次元の運動の運動方程式を解く

質量 m の質点が、 x 軸上の運動をしている場合を考える。時刻 t の瞬間の質点の加速度を $ax(t)$ 、質点の作用している力を $f(t)$ とすると、ニュートン $(Newton)$ の運動方程式は次式のようになる。

$$ma(t) = f(t) \tag{3.1}$$

時刻 t の瞬間の質点の速度を $vx(t)$ 、位置を $x(t)$ とする。また、質点に作用している力 $f(t)$ は一般的に、時刻 t 、速度 $vx(t)$ 、位置 $x(t)$ の関数である。

$$a(t) = \frac{f\Big(t, vx\left(t\right), x\left(t\right)\Big)}{m} \tag{3.2}$$

$$a(t) = \frac{f\Big(t, vx\left(t\right), x\left(t\right)\Big)}{m} \tag{3.3}$$

$$\frac{d^2}{dt^2}x(t) = \frac{f\Big(t, \frac{d}{dt}x\left(t\right), x\left(t\right)\Big)}{m} \tag{3.4}$$

この２階の１元微分方程式を、次式のように書き換えることにより１階の２元微分方程式にする。

$$\begin{cases} \dfrac{d}{dt}vx(t) = \dfrac{f\Big(t, vx\left(t\right), x\left(t\right)\Big)}{m} \\ \dfrac{d}{dt}x(t) = vx(t) \end{cases} \tag{3.5}$$

3.1　１次元の物理現象１ - 単振動

位置 x に比例する引力 $-kx$ を受ける質点の運動を考える。

$$m\frac{d^2}{dt^2}x = -kx \tag{3.6}$$

ここで、物理定数 ω_0 を使って書き換えると次のようになる。

$$\frac{d^2}{dt^2}x + \omega_0^2 x = 0 \tag{3.7}$$

$$\omega_0 = \sqrt{\frac{k}{m}} \tag{3.8}$$

この単振動のニュートンの運動方程式は、２階の１元微分方程式なので、１階の２元微分方程式に書き換える。

$$\begin{cases} \dfrac{d}{dt}vx(t) = -\omega_0^2 x = fax(x) \\ \dfrac{d}{dt}x(t) = vx \end{cases} \tag{3.9}$$

3.1.1　解析解を求める。

同次定数係数の２階線形微分方程式

$$\frac{d^2}{dt^2}x(t) + \omega_0^2 x(t) = 0 \tag{3.10}$$

ここで、 $x(t) = e^{qt}$ とおく。

$$\left(q^2 + \omega_0^2\right) e^{qt} = 0 \tag{3.11}$$

特性方程式： $q^2 + \omega_0^2 = 0$

$$q = 0 \pm i\omega_0 \tag{3.12}$$

よって、特性方程式は、２つの虚数解 $q_1 \pm iq_2$ を持つので、一般解は次のようになる。

$$x(t) = e^{q_1 t}\Big\{C_1 \sin(q_2 t) + C_2 \cos(q_2 t)\Big\} \tag{3.13}$$

$$x(t) = e^{0t}\Big\{C_1 \sin(\omega_0 t) + C_2 \cos(\omega_0 t)\Big\} \tag{3.14}$$

$$x(t) = C_1 \sin(\omega_0 t) + C_2 \cos(\omega_0 t) \tag{3.15}$$

3.1.2　数値解を求め、解析解と比較する。

初期条件 ： $t = 0$ のとき、 $vx(0) = 0.0$ 、 $x(0) = 1.0$ とする。また、物理定数 ： $\omega_0 = 1.0$ とする。

```
//C++の標準入出力関数ヘッダファイルをインクルードする。
#include<iostream.h>
//C++のファイル入出力関数ヘッダファイルをインクルードする。
#include <fstream.h>
//数学関数ヘッダファイルをインクルードする。
#include<math.h>

//物理定数 w0 を定義する。
#define w0 1.0
//刻み幅 dt を定義する。
#define dt 0.01
//時間 t の初期値を定義する。
#define initial_t 0.0
//位置 x の初期値を定義する。
#define initial_x 1.0
//速度 vx の初期値を定義する。
#define initial_vx 0.0
//任意定数 C1 を定義する。
#define C1 ( initial_x * sin( w0 * initial_t ) + initial_vx * cos( w0 * initial_t ) / w0 )
//任意定数 C2 を定義する。
#define C2 ( initial_x * cos( w0 * initial_t ) - initial_vx * sin( w0 * initial_t ) / w0 )

//ファイル出力のためのファイル名を決め、ファイル出力のためのファイルを開く。
ofstream fileout( "itizigen_1.dat" , ios::out );

//関数 fax : 加速度
double fax( double x )
{
    return ( - w0 * w0 * x );
}
```

```
//関数 runge ( 引数 : &vx と &x , 戻り値  : なし )
void runge( double &vx , double &x )
{
    //配列 kvx と kx の変数宣言
    double kvx[4] , kx[4];

    kvx[0] = fax( x ) * dt;
    kx[0] = ( vx ) * dt;
    kvx[1] = fax( x + kx[0] / 2.0 ) * dt;
    kx[1] = ( vx + kvx[0] / 2.0 ) * dt;
    kvx[2] = fax( x + kx[1] / 2.0 ) * dt;
    kx[2] = ( vx + kvx[1] / 2.0 ) * dt;
    kvx[3] = fax( x + kx[2] ) * dt;
    kx[3] = ( vx + kvx[2] ) * dt;

    x = x + ( kx[0] + 2.0 * kx[1] + 2.0 * kx[2] + kx[3] ) / 6.0;
    vx = vx + ( kvx[0] + 2.0 * kvx[1] + 2.0 * kvx[2] + kvx[3] ) / 6.0;
}

//main 関数
int main()
{
    //変数 t と vx と x と ft_vx と ft_x の変数宣言
    double t, vx, x, ft_vx, ft_x;

    //時間 t の初期値を代入する。
    t = initial_t;
    //速度 vx の初期値を代入する。
    vx = initial_vx;
    //位置 x の初期値を代入する。
    x = initial_x;

    //while 文
    while( t <= 100.0 ){
        //速度 vx の解析解を求める。
        ft_vx = C1 * w0 * cos( w0 * t ) - C2 * w0 * sin( w0 * t );
        //位置 x の解析解を求める。
        ft_x = C1 * sin( w0 * t ) + C2 * cos( w0 * t );

        //ファイル出力で解析解と数値解の比較を行なう。
        fileout << t << '\t' << ft_vx << '\t' << vx << '\t' << ft_x << '\t' << x << '\n';

        //関数 runge の呼び出し。
        runge( vx , x );

        t = t + dt;
    }

    //ファイル出力のためのファイルを閉じる。
    fileout.close();

    return ( 0 );
}
```

3.2　１次元の物理現象２ - 減衰振動

位置 x に比例する引力 $-kx$ と 速度に比例する抵抗力 $-\mu\frac{d}{dt}x$ を受ける質点の運動を考える。

$$m\frac{d^2}{dt^2}x = -kx - \mu\frac{d}{dt}x \tag{3.16}$$

ここで、物理定数 ω_0 と γ を使って書き換えると次のようになる。

$$\frac{d^2}{dt^2}x + 2\gamma\frac{d}{dt}x + \omega_0^2 x = 0 \tag{3.17}$$

$$\begin{cases} \omega_0 = \sqrt{\dfrac{k}{m}} \\ \gamma = \dfrac{\mu}{2m} \end{cases} \tag{3.18}$$

この減衰振動のニュートンの運動方程式は、２階の１元微分方程式なので、１階の２元微分方程式に書き換える。

$$\begin{cases} \dfrac{d}{dt}vx(t) = -2\gamma vx - \omega_0^2 x = fax(vx, x) \\ \dfrac{d}{dt}x(t) = vx \end{cases} \tag{3.19}$$

3.2.1 解析解を求める。

同次定数係数の２階線形微分方程式

$$\frac{d^2}{dt^2}x(t) + 2\gamma\frac{d}{dt}x(t) + \omega_0^2 x(t) = 0 \tag{3.20}$$

ここで、 $x(t) = e^{qt}$ とおく。

$$\left(q^2 + 2\gamma q + \omega_0^2\right) e^{qt} = 0 \tag{3.21}$$

$\gamma > \omega_0$ のときの特性方程式： $q^2 + 2\gamma q + \omega_0^2 = 0$

$$q = -\gamma \pm \sqrt{\gamma^2 - \omega_0^2} \tag{3.22}$$

よって、特性方程式は、２つの実数解 q_1, q_2 を持つので、一般解は次のようになる。

$$x(t) = C_1 e^{q_1 t} + C_2 e^{q_2 t} \tag{3.23}$$

$$x(t) = C_1 e^{\left(-\gamma + \sqrt{\gamma^2 - \omega_0^2}\right)t} + C_2 e^{\left(-\gamma - \sqrt{\gamma^2 - \omega_0^2}\right)t} \tag{3.24}$$

$\gamma = \omega_0$ のときの特性方程式： $q^2 + 2\gamma q + \gamma^2 = 0$

$$q = -\gamma \tag{3.25}$$

よって、特性方程式は、重解 q を持つので、一般解は次のようになる。

$$x(t) = (C_1 + C_2 t)\, e^{qt} \tag{3.26}$$

$$x(t) = (C_1 + C_2 t)\, e^{-\gamma t} \tag{3.27}$$

$\gamma < \omega_0$ のときの特性方程式： $q^2 + 2\gamma q + \omega_0^2 = 0$

$$q = -\gamma \pm i\sqrt{\omega_0^2 - \gamma^2} \tag{3.28}$$

よって、特性方程式は、２つの虚数解 $q_1 \pm iq_2$ を持つので、一般解は次のようになる。

$$x(t) = e^{q_1 t}\Big\{C_1 \sin(q_2 t) + C_2 \cos(q_2 t)\Big\} \tag{3.29}$$

$$x(t) = e^{-\gamma t}\left\{C_1 \sin\left(\sqrt{\omega_0^2 - \gamma^2}t\right) + C_2 \cos\left(\sqrt{\omega_0^2 - \gamma^2}t\right)\right\} \tag{3.30}$$

3.2.2 数値解を求め、解析解と比較する。

初期条件 ： $t = 0$ のとき、 $vx(0) = 0.0$ 、 $x(0) = 1.0$ とする。また、物理定数 ： $\omega_0 = 1.0$ 、 $\gamma = 1.2$ とする。

```
//C++の標準入出力関数ヘッダファイルをインクルードする。
#include<iostream.h>
//C++のファイル入出力関数ヘッダファイルをインクルードする。
#include <fstream.h>
//数学関数ヘッダファイルをインクルードする。
#include<math.h>

//物理定数 w0 を定義する。
#define w0 1.0
//物理定数 r を定義する。
#define r 1.2
//刻み幅 dt を定義する。
#define dt 0.01
//時間 t の初期値を定義する。
#define initial_t 0.0
//位置 x の初期値を定義する。
#define initial_x 1.0
//速度 vx の初期値を定義する。
#define initial_vx 0.0
//任意定数 C1 を定義する。
#define C1 ( - exp( ( r + sqrt( r * r - w0 * w0 ) ) * initial_t ) * ( initial_vx + ( r - sqrt( r * r - w0 * w0 ) ) * initial_x )
            / ( 2.0 * sqrt( r * r - w0 * w0 ) ) )
//任意定数 C2 を定義する。
#define C2 ( exp( ( r - sqrt( r * r - w0 * w0 ) ) * initial_t ) * ( initial_vx + ( r + sqrt( r * r - w0 * w0 ) ) * initial_x )
            / ( 2.0 * sqrt( r * r - w0 * w0 ) ) )

//ファイル出力のためのファイル名を決め、ファイル出力のためのファイルを開く。
ofstream fileout( "itizigen_2.dat" , ios::out );

//関数 fax : 加速度
double fax( double vx , double x )
{
    return ( - 2.0 * r * vx - w0 * w0 * x );
}

//関数 runge ( 引数 : &vx と &x , 戻り値  : なし )
void runge( double &vx , double &x )
{
    //配列 kvx と kx の変数宣言
    double kvx[4] , kx[4];

    kvx[0] = fax( vx , x ) * dt;
    kx[0] = ( vx ) * dt;
    kvx[1] = fax( vx + kvx[0] / 2.0 , x + kx[0] / 2.0 ) * dt;
    kx[1] = ( vx + kvx[0] / 2.0 ) * dt;
    kvx[2] = fax( vx + kvx[1] / 2.0 , x + kx[1] / 2.0 ) * dt;
    kx[2] = ( vx + kvx[1] / 2.0 ) * dt;
    kvx[3] = fax( vx + kvx[2] , x + kx[2] ) * dt;
    kx[3] = ( vx + kvx[2] ) * dt;

    x = x + ( kx[0] + 2.0 * kx[1] + 2.0 * kx[2] + kx[3] ) / 6.0;
    vx = vx + ( kvx[0] + 2.0 * kvx[1] + 2.0 * kvx[2] + kvx[3] ) / 6.0;
}

//main 関数
int main()
{
    //変数 t と vx と x と ft_vx と ft_x の変数宣言
    double t, vx, x, ft_vx, ft_x;

    //時間 t の初期値を代入する。
    t = initial_t;
    //速度 vx の初期値を代入する。
    vx = initial_vx;
    //位置 x の初期値を代入する。
    x = initial_x;

    //while 文
    while( t <= 100.0 ){
        //速度 vx の解析解を求める。
        ft_vx = C1 * ( - r - sqrt( r * r - w0 * w0 ) ) * exp( ( - r - sqrt( r * r - w0 * w0 ) ) * t )
                + C2 * ( - r + sqrt( r * r - w0 * w0 ) ) * exp( ( - r + sqrt( r * r - w0 * w0 ) ) * t );
        //位置 x の解析解を求める。
        ft_x = C1 * exp( ( - r - sqrt( r * r - w0 * w0 ) ) * t ) + C2 * exp( ( - r + sqrt( r * r - w0 * w0 ) ) * t );

        //ファイル出力で解析解と数値解の比較を行なう。
        fileout << t << '\t' << ft_vx << '\t' << vx << '\t' << ft_x << '\t' << x << '\n';

        //関数 runge の呼び出し。
```

```
        runge( vx , x );

        t = t + dt;
    }

    //ファイル出力のためのファイルを閉じる。
    fileout.close();

    return ( 0 );
}
```

3.3 １次元の物理現象３ - 無減衰系の強制振動

位置 x に比例する引力 $-kx$ と 周期的外力 $a\sin(pt)$ を受ける質点の運動を考える。

$$m\frac{d^2}{dt^2}x = -kx + a\sin(pt) \tag{3.31}$$

ここで、物理定数 ω_0 と f_0 を使って書き換えると次のようになる。

$$\frac{d^2}{dt^2}x + \omega_0^2 x = f_0\sin(pt) \tag{3.32}$$

$$\begin{cases} \omega_0 = \sqrt{\dfrac{k}{m}} \\ f_0 = \dfrac{a}{m} \end{cases} \tag{3.33}$$

この無減衰系の強制振動のニュートンの運動方程式は、２階の１元微分方程式なので、１階の２元微分方程式に書き換える。

$$\begin{cases} \dfrac{d}{dt}vx(t) = -\omega_0^2 x + f_0\sin(pt) = fax(t,x) \\ \dfrac{d}{dt}x(t) = vx \end{cases} \tag{3.34}$$

3.3.1 解析解を求める。

非同次定数係数の２階線形微分方程式

$$\frac{d^2}{dt^2}x(t) + \omega_0^2 x(t) = f_0\sin(pt) \tag{3.35}$$

同次方程式の一般解を求める。

同次定数係数の２階線形微分方程式

$$\frac{d^2}{dt^2}x(t) + \omega_0^2 x(t) = 0 \tag{3.36}$$

これは、単振動のときの一般解（3.15）と同じになる。

$$x(t) = C_1 \sin(\omega_0 t) + C_2 \cos(\omega_0 t) \tag{3.37}$$

非同次方程式の特殊解を求める。

非同次定数係数の 2 階線形微分方程式

$$\frac{d^2}{dt^2}x(t) + \omega_0^2 x(t) = f_0 \sin(pt) \tag{3.38}$$

$\omega_0 \neq p$ のとき

$$\frac{d^2}{dt^2}x(t) + \omega_0^2 x(t) = f_0 \sin(pt) \tag{3.39}$$

ここで、 $x(t) = A\sin(pt) + B\cos(pt)$ とおく。

$$\left(-Ap^2 + A\omega_0^2 - f_0\right)\sin(pt) + \left(-Bp^2\right)\cos(pt) = 0 \tag{3.40}$$

$$\begin{cases} A = \dfrac{f_0}{\omega_0^2 - p^2} \\ B = 0 \end{cases} \tag{3.41}$$

よって、 $\omega_0 \neq p$ のとき、特殊解は次のようになる。

$$x(t) = \frac{f_0}{\omega_0^2 - p^2}\sin(pt) \tag{3.42}$$

$\omega_0 = p$ のとき

$$\frac{d^2}{dt^2}x(t) + \omega_0^2 x(t) = f_0 \sin(\omega_0 t) \tag{3.43}$$

ここで、 $x(t) = t\left(A\sin(\omega_0 t) + B\cos(\omega_0 t)\right)$ とおく。

$$\left(-A\omega_0^2 t - 2B\omega_0 + A\omega_0^2 t - f_0\right)\sin(pt) + \left(-B\omega_0^2 t + 2A\omega_0 + B\omega_0^2 t\right)\cos(pt) = 0 \tag{3.44}$$

$$\begin{cases} A = 0 \\ B = -\dfrac{f_0}{2\omega_0} \end{cases} \tag{3.45}$$

よって、 $\omega_0 = p$ のとき、特殊解は次のようになる。

$$x(t) = -\frac{f_0}{2\omega_0} t \cos(\omega_0 t) \tag{3.46}$$

非同次方程式の一般解を求める。

非同次方程式の一般解 は、 同次方程式の一般解 と 非同次方程式の特殊解 の和になる。
$\omega_0 \neq p$ のとき

$$x(t) = C_1 \sin(\omega_0 t) + C_2 \cos(\omega_0 t) + \frac{f_0}{\omega_0^2 - p^2} \sin(pt) \tag{3.47}$$

$\omega_0 = p$ のとき

$$x(t) = C_1 \sin(\omega_0 t) + C_2 \cos(\omega_0 t) - \frac{f_0}{2\omega_0} t \cos(\omega_0 t) \tag{3.48}$$

3.3.2 数値解を求め、解析解と比較する。

初期条件 ： $t = 0$ のとき、 $vx(0) = 0.0$ 、 $x(0) = 1.0$ とする。また、物理定数 ： $\omega_0 = 1.0$ 、 $f_0 = 1.2$ 、 $p = 1.4$ とする。

```
//C++の標準入出力関数ヘッダファイルをインクルードする。
#include<iostream.h>
//C++のファイル入出力関数ヘッダファイルをインクルードする。
#include <fstream.h>
//数学関数ヘッダファイルをインクルードする。
#include<math.h>

//物理定数 w0 を定義する。
#define w0 1.0
//物理定数 f0 を定義する。
#define f0 1.2
//物理定数 p を定義する。
#define p 1.4
//刻み幅 dt を定義する。
#define dt 0.01
//時間 t の初期値を定義する。
#define initial_t 0.0
//位置 x の初期値を定義する。
#define initial_x 1.0
//速度 vx の初期値を定義する。
#define initial_vx 0.0
//任意定数 C1 を定義する。
#define C1 ( ( ( initial_vx * ( p * p - w0 * w0 ) + f0 * p * cos( p * initial_t ) ) * cos( w0 * initial_t )
             + w0 * ( ( p * p - w0 * w0 ) * initial_x + f0 * sin( p * initial_t ) ) * sin( w0 * initial_t ) )
             / ( w0 * ( p * p - w0 * w0 ) ) )
//任意定数 C2 を定義する。
#define C2 ( ( ( initial_vx * ( p * p - w0 * w0 ) + f0 * p * cos( p * initial_t ) ) * sin( w0 * initial_t )
             + w0 * ( ( w0 * w0 - p * p ) * initial_x - f0 * sin( p * initial_t ) ) * cos( w0 * initial_t ) )
             / ( w0 * ( w0 * w0 - p * p ) ) )

//ファイル出力のためのファイル名を決め、ファイル出力のためのファイルを開く。
ofstream fileout( "itizigen_3.dat" , ios::out );

//関数 fax : 加速度
double fax( double t , double x )
{
    return ( - w0 * w0 * x + f0 * sin( p * t ) );
}

//関数 runge ( 引数 : t と &vx と &x , 戻り値  : なし )
void runge( double t , double &vx , double &x )
{
    //配列 kvx と kx の変数宣言
    double kvx[4] , kx[4];

    kvx[0] = fax( t , x ) * dt;
    kx[0] = ( vx ) * dt;
```

```
    kvx[1] = fax( t + dt / 2.0 , x + kx[0] / 2.0 ) * dt;
    kx[1] = ( vx + kvx[0] / 2.0 ) * dt;
    kvx[2] = fax( t + dt / 2.0 , x + kx[1] / 2.0 ) * dt;
    kx[2] = ( vx + kvx[1] / 2.0 ) * dt;
    kvx[3] = fax( t + dt , x + kx[2] ) * dt;
    kx[3] = ( vx + kvx[2] ) * dt;

    x = x + ( kx[0] + 2.0 * kx[1] + 2.0 * kx[2] + kx[3] ) / 6.0;
    vx = vx + ( kvx[0] + 2.0 * kvx[1] + 2.0 * kvx[2] + kvx[3] ) / 6.0;
}

//main 関数
int main()
{
    //変数 t と vx と x と ft_vx と ft_x の変数宣言
    double t, vx, x, ft_vx, ft_x;

    //時間 t の初期値を代入する。
    t = initial_t;
    //速度 vx の初期値を代入する。
    vx = initial_vx;
    //位置 x の初期値を代入する。
    x = initial_x;

    //while 文
    while( t <= 100.0 ){
        //速度 vx の解析解を求める。
        ft_vx = C1 * w0 * cos( w0 * t ) - C2 * w0 * sin( w0 * t ) + f0 * p * cos( p * t ) / ( w0 * w0 - p * p );
        //位置 x の解析解を求める。
        ft_x = C1 * sin( w0 * t ) + C2 * cos( w0 * t ) + f0 * sin( p * t ) / ( w0 * w0 - p * p );

        //ファイル出力で解析解と数値解の比較を行なう。
        fileout << t << '\t' << ft_vx << '\t' << vx << '\t' << ft_x << '\t' << x << '\n';

        //関数 runge の呼び出し。
        runge( t , vx , x );

        t = t + dt;
    }

    //ファイル出力のためのファイルを閉じる。
    fileout.close();

    return ( 0 );
}
```

3.4 １次元の物理現象４ - 減衰系の強制振動

位置 x に比例する引力 $-kx$ と 速度に比例する抵抗力 $-\mu\frac{d}{dt}x$ と 周期的外力 $a\sin(pt)$ を受ける質点の運動を考える。

$$m\frac{d^2}{dt^2}x = -kx - \mu\frac{d}{dt}x + a\sin(pt) \tag{3.49}$$

ここで、物理定数 ω_0 と γ と f_0 を使って書き換えると次のようになる。

$$\frac{d^2}{dt^2}x + 2\gamma\frac{d}{dt}x + \omega_0^2 x = f_0\sin(pt) \tag{3.50}$$

$$\begin{cases} \omega_0 = \sqrt{\dfrac{k}{m}} \\ \gamma = \dfrac{\mu}{2m} \\ f_0 = \dfrac{a}{m} \end{cases} \tag{3.51}$$

この減衰系の強制振動のニュートンの運動方程式は、２階の１元微分方程式なので、１階の２元微分方程式に書き換える。

$$\begin{cases} \dfrac{d}{dt}vx(t) = -2\gamma vx - \omega_0^2 x + f_0 \sin(pt) = fax(t, vx, x) \\ \dfrac{d}{dt}x(t) = vx \end{cases} \tag{3.52}$$

3.4.1　解析解を求める。

非同次定数係数の２階線形微分方程式

$$\frac{d^2}{dt^2}x(t) + 2\gamma\frac{d}{dt}x(t) + \omega_0^2 x(t) = f_0 \sin(pt) \tag{3.53}$$

同次方程式の一般解を求める。

同次定数係数の２階線形微分方程式

$$\frac{d}{dt}vx(t) = -2\gamma vx - \omega_0^2 x + f_0 \sin(pt) = 0 \tag{3.54}$$

これは、減衰振動のときの一般解（3.24）、（3.27）、（3.30）と同じになる。
$\gamma > \omega_0$ のとき

$$x(t) = C_1 e^{\left(-\gamma + \sqrt{\gamma^2 - \omega_0^2}\right)t} + C_2 e^{\left(-\gamma - \sqrt{\gamma^2 - \omega_0^2}\right)t} \tag{3.55}$$

$\gamma = \omega_0$ のとき

$$x(t) = (C_1 + C_2 t)\, e^{\gamma t} \tag{3.56}$$

$\gamma < \omega_0$ のとき

$$x(t) = e^{-\gamma t}\left\{ C_1 \sin\left(\sqrt{\omega_0^2 - \gamma^2}\, t\right) + C_2 \cos\left(\sqrt{\omega_0^2 - \gamma^2}\, t\right) \right\} \tag{3.57}$$

非同次方程式の特殊解を求める。

非同次定数係数の２階線形微分方程式

$$\frac{d^2}{dt^2}x(t) + 2\gamma\frac{d}{dt}x + \omega_0^2 x(t) = f_0 \sin(pt) \tag{3.58}$$

ここで、 $x(t) = A\sin(pt) + B\cos(pt)$ とおく。

$$\left(-Ap^2 - 2B\gamma p + A\omega_0^2 - f_0\right)\sin(pt) + \left(-Bp^2 + 2A\gamma p + B\omega_0^2\right)\cos(pt) = 0 \tag{3.59}$$

$$\begin{cases} A = \dfrac{f_0}{\left(\omega_0^2 - p^2\right)^2 + 4\gamma^2 p^2}\left(\omega_0^2 - p^2\right) \\ B = -\dfrac{f_0}{\left(\omega_0^2 - p^2\right)^2 + 4\gamma^2 p^2} 2\gamma p \end{cases} \tag{3.60}$$

よって、特殊解は次のようになる。

$$x(t) = \frac{f_0}{\left(\omega_0^2 - p^2\right)^2 + 4\gamma^2 p^2}\Big\{\left(\omega_0^2 - p^2\right)\sin(pt) - 2\gamma p\cos(pt)\Big\} \tag{3.61}$$

非同次方程式の一般解を求める。

非同次方程式の一般解 は、 同次方程式の一般解 と 非同次方程式の特殊解 の和になる。
$\gamma > \omega_0$ のとき

$$x(t) = C_1 e^{\left(-\gamma + \sqrt{\gamma^2 - \omega_0^2}\right)t} + C_2 e^{\left(-\gamma - \sqrt{\gamma^2 - \omega_0^2}\right)t} + \frac{f_0}{\left(\omega_0^2 - p^2\right)^2 + 4\gamma^2 p^2}\Big\{\left(\omega_0^2 - p^2\right)\sin(pt) - 2\gamma p\cos(pt)\Big\} \tag{3.62}$$

$\gamma = \omega_0$ のとき

$$x(t) = \left(C_1 + C_2 t\right) e^{\gamma t} + \frac{f_0}{\left(\omega_0^2 - p^2\right)^2 + 4\gamma^2 p^2}\Big\{\left(\omega_0^2 - p^2\right)\sin(pt) - 2\gamma p\cos(pt)\Big\} \tag{3.63}$$

$\gamma < \omega_0$ のとき

$$x(t) = e^{-\gamma t}\left\{C_1 \sin\left(\sqrt{\omega_0^2 - \gamma^2}\, t\right) + C_2 \cos\left(\sqrt{\omega_0^2 - \gamma^2}\, t\right)\right\} + \frac{f_0}{\left(\omega_0^2 - p^2\right)^2 + 4\gamma^2 p^2}\Big\{\left(\omega_0^2 - p^2\right)\sin(pt) - 2\gamma p\cos(pt)\Big\} \tag{3.64}$$

3.4.2 数値解を求め、解析解と比較する。

初期条件 ： $t = 0$ のとき、 $vx(0) = 0.0$ 、 $x(0) = 1.0$ とする。また、物理定数 ： $\omega_0 = 1.0$ 、 $\gamma = 1.2$ 、 $f_0 = 1.4$ 、 $p = 1.6$ とする。

```
//C++の標準入出力関数ヘッダファイルをインクルードする。
#include<iostream.h>
//C++のファイル入出力関数ヘッダファイルをインクルードする。
#include <fstream.h>
//数学関数ヘッダファイルをインクルードする。
#include<math.h>

//物理定数 w0 を定義する。
#define w0 1.0
//物理定数 r を定義する。
#define r 1.2
//物理定数 f0 を定義する。
#define f0 1.4
//物理定数 p を定義する。
#define p 1.6
//刻み幅 dt を定義する。
#define dt 0.01
//時間 t の初期値を定義する。
#define initial_t 0.0
//位置 x の初期値を定義する。
#define initial_x 1.0
//速度 vx の初期値を定義する。
#define initial_vx 0.0
//任意定数 C1 を定義する。
#define C1 ( exp( ( r + sqrt( r * r - w0 * w0 ) ) * initial_t ) * ( - ( 4.0 * r * r * p * p + ( p * p - w0 * w0 ) * ( p * p - w0 * w0 ) )
            * ( initial_vx + ( r - sqrt( r * r - w0 * w0 ) ) * initial_x ) + f0 * p * ( - 2.0 * r * r - p * p + w0 * w0 + 2.0 * r
            * sqrt( r * r - w0 * w0 ) ) * cos( p * initial_t ) + f0 * ( sqrt( r * r - w0 * w0 ) * ( p * p - w0 * w0 ) + r * ( p * p + w0 * w0 ) )
            * sin( p * initial_t ) ) ) / ( 2.0 * sqrt( r * r - w0 * w0 ) * ( 4.0 * r * r * p * p + ( p * p - w0 * w0 ) * ( p * p - w0 * w0 ) ) )
//任意定数 C2 を定義する。
#define C2 ( exp( ( r - sqrt( r * r - w0 * w0 ) ) * initial_t ) * ( ( 4.0 * r * r * p * p + ( p * p - w0 * w0 ) * ( p * p - w0 * w0 ) )
            * ( initial_vx + ( r + sqrt( r * r - w0 * w0 ) ) * initial_x ) + f0 * p * ( 2.0 * r * r + p * p - w0 * w0 + 2.0 * r
            * sqrt( r * r - w0 * w0 ) ) * cos( p * initial_t ) - f0 * ( sqrt( r * r - w0 * w0 ) * ( - p * p + w0 * w0 ) + r * ( p * p + w0 * w0 ) )
```

```
            * sin( p * initial_t ) ) ) / ( 2.0 * sqrt( r * r - w0 * w0 ) * ( 4.0 * r * r * p * p + ( p * p - w0 * w0 ) * ( p * p - w0 * w0 ) ) )

//ファイル出力のためのファイル名を決め、ファイル出力のためのファイルを開く。
ofstream fileout( "itizigen_4.dat" , ios::out );

//関数 fax : 加速度
double fax( double t , double vx , double x )
{
    return ( - 2.0 * r * vx - w0 * w0 * x + f0 * sin( p * t ) );
}

//関数 runge ( 引数 : t と &vx と &x , 戻り値  : なし )
void runge( double t , double &vx , double &x )
{
    //配列 kvx と kx の変数宣言
    double kvx[4] , kx[4];

    kvx[0] = fax( t , vx , x ) * dt;
    kx[0] = ( vx ) * dt;
    kvx[1] = fax( t + dt / 2.0 , vx + kvx[0] / 2.0 , x + kx[0] / 2.0 ) * dt;
    kx[1] = ( vx + kvx[0] / 2.0 ) * dt;
    kvx[2] = fax( t + dt / 2.0 , vx + kvx[1] / 2.0 , x + kx[1] / 2.0 ) * dt;
    kx[2] = ( vx + kvx[1] / 2.0 ) * dt;
    kvx[3] = fax( t + dt , vx + kvx[2] , x + kx[2] ) * dt;
    kx[3] = ( vx + kvx[2] ) * dt;

    x = x + ( kx[0] + 2.0 * kx[1] + 2.0 * kx[2] + kx[3] ) / 6.0;
    vx = vx + ( kvx[0] + 2.0 * kvx[1] + 2.0 * kvx[2] + kvx[3] ) / 6.0;
}

//main 関数
int main()
{
    //変数 t と vx と x と ft_vx と ft_x の変数宣言
    double t, vx, x, ft_vx, ft_x;

    //時間 t の初期値を代入する。
    t = initial_t;
    //速度 vx の初期値を代入する。
    vx = initial_vx;
    //位置 x の初期値を代入する。
    x = initial_x;

    //while 文
    while( t <= 100.0 ){
        //速度 vx の解析解を求める。
        ft_vx = C1 * ( - r - sqrt( r * r - w0 * w0 ) ) * exp( ( - r - sqrt( r * r - w0 * w0 ) ) * t )
                + C2 * ( - r + sqrt( r * r - w0 * w0 ) ) * exp( ( - r + sqrt( r * r - w0 * w0 ) ) * t )
                + f0 * ( 2.0 * r * p * p * sin( p * t ) - ( p * p - w0 * w0 ) * p * cos( p * t ) )
                / ( 4.0 * r * r * p * p + ( p * p - w0 * w0 ) * ( p * p - w0 * w0 ) );
        //位置 x の解析解を求める。
        ft_x = C1 * exp( ( - r - sqrt( r * r - w0 * w0 ) ) * t )
               + C2 * exp( ( - r + sqrt( r * r - w0 * w0 ) ) * t )
               - f0 * ( 2.0 * r * p * cos( p * t ) + ( p * p - w0 * w0 ) * sin( p * t ) )
               / ( 4.0 * r * r * p * p + ( p * p - w0 * w0 ) * ( p * p - w0 * w0 ) );

        //ファイル出力で解析解と数値解の比較を行なう。
        fileout << t << '\t' << ft_vx << '\t' << vx << '\t' << ft_x << '\t' << x << '\n';

        //関数 runge の呼び出し。
        runge( t , vx , x );

        t = t + dt;
    }

    //ファイル出力のためのファイルを閉じる。
    fileout.close();

    return ( 0 );
}
```