5. 基本的な力学問題

2017/11/6

1 Runge-Kutta 法

前回の講義では Euler 法を用いて、一様重力中の落下に対する微分方程式を数値的に解きました。しかし、Euler 法は、精度と安定性に課題があります。Runge-Kutta 法は、非常に広く用いられている標準的な一階連立微分方程式の数値解法です。

独立変数を t、従属変数を $\vec{y}$ とします。$\vec{y}$ の成分を $y_j (0 \leq j < n)$ とします。このとき、y_j に対する連立微分方程式が

$$\frac{\mathrm{d}y_j}{\mathrm{d}t} = f_j\left(t, \vec{y}\right) \tag{1.1}$$

で与えられているとしましょう。ある点 t と $\vec{y}(t)$ が与えられている時、$\vec{y}(t+h)$ を以下のように求めるのが (4 次の) Runge-Kutta 法です。

$$\begin{aligned}
\vec{k}_1 &= h\vec{f}\left(t, \vec{y}\right) \\
\vec{k}_2 &= h\vec{f}\left(t + \frac{1}{2}h, \vec{y} + \frac{1}{2}\vec{k}_1\right) \\
\vec{k}_3 &= h\vec{f}\left(t + \frac{1}{2}h, \vec{y} + \frac{1}{2}\vec{k}_2\right) \\
\vec{k}_4 &= h\vec{f}\left(t + h, \vec{y} + \vec{k}_3\right) \\
\vec{y}(t+h) &= \vec{y}(t) + \frac{1}{6}\left(\vec{k}_1 + 2\vec{k}_2 + 2\vec{k}_3 + \vec{k}_4\right) + O\left(h^5\right)
\end{aligned} \tag{1.2}$$

この方法では、$O(h^4)$ までの精度で数値計算をうことができます。

2　調和振動子

Runge-Kutta 法で調和振動子の運動を数値的に求めましょう。調和振動子は、その位置 x に対する

$$m\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = -kx \tag{2.1}$$

という二階微分方程式で記述されます。m は振動子についた錘の質量、k はバネ定数です。この微分方程式の一般解は、A と B を積分定数として

$$x = A\cos(\omega t) + B\sin(\omega t) \tag{2.2}$$

と表すことができます。ここで $\omega^2 = k/m$ です。対応する速度は

$$v = -A\omega\sin(\omega t) + B\omega\cos(\omega t) \tag{2.3}$$

です。

課題 1　数値解を求める準備として、前回作成したプロジェクト `DifferentialEquation` の下に新しいパッケージ `oscillators` を作成し、配布された新しいクラスファイル `HarmonicOscillator.java` を作成し、コンストラクタ中に対応する微分方程式を記載しなさい。

課題 2　`HarmonicOscillator` クラスは、`myLib.rungeKutta.Dynamics` の拡張クラスとして定義されています。`myLib.rungeKutta.Dynamics` クラスのコンストラクタが行っていることを確認しなさい。

課題 3　このクラスを実行すると、ファイル `HarmonicOscillator-output.txt` に変位の時間変化が出力されます。この出力を、ソースファイル 2.1 を使ってプロットし、結果を確かめなさい。

ソースコード 2.1　`harmonic.plt`

```
1 set terminal png 14
2 set xlabel "t"
3 set ylabel "x"
4 set yrange [−1.5:1.5]
5 set xrange [0:20]
6 set ytic 0.5
```

```
7  set title "Harmonic␣Oscillator"
8  set output "HarmonicOscillator.png"
9  B=1
10 W=1
11 plot "HarmonicOscillator-output.txt", B*sin(W*x) lw 3
```

課題 4　`main()` を見ると、課題 3 では、$\omega = 1$ として、初期値として $x(0) = 0$ と $v(0) = 1$ に対するシミュレーションを行っていることが分かります。$x(0) = 0$ であることから、式 (2.2) より、$A = 0$ となります。さらに、$v(0) = 1$ であることから、式 (2.3) より、$B = 1$ となります。そのため、Program 2.1 では、数値計算の結果と $\sin(x)$ を比較しています。

それでは、初期値が $x(0) = 1$ と $v(0) = 1$ の場合はどうでしょう。その解を導くとともに、数値計算と厳密解とを、図を描いて比較しなさい。

3　減衰振動

運動方程式の右辺に速度に比例した項を入れることで摩擦を表すことができます。

$$m\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = -kx - \gamma v \tag{3.1}$$

同じパッケージ `oscillators` に、調和振動子のクラス `HarmonicOscillator` のコピーから減衰振動のクラス `DumpedOscillator` を作成します。

課題 5　`DumpedOscillator` のコンストラクタに、式 (3.1) に対応した微分方程式を記述しなさい。その際に、パラメタ γ が増えていることに注意しなさい。`main()` も対応して変更しなさい。

課題 6　減衰振動を `gnuplot` で作図するためのスクリプト `dumped.plt` を、ソースファイル 2.1 を参考に作成しなさい。画像の出力先ファイル名は `DumpedOscillator.png` としなさい。

課題 7　スクリプト `dumped.plt` を用いて、変位の時間変化を出力し、厳密解と比較しなさい。講義で示したように振幅は $e^{-\alpha t}$ で減衰します。この減衰曲線も、ともに描画しなさい。

3.1 発展問題

時間に余裕のある者は以下の課題も回答しましょう。

時間発展を追う際に、Dynamics クラス (減衰振動のクラス DumpedOscillator の親クラス) のメソッド

```
public List<Point2D.Double> evolution(double h, int nstep)
```

を用いました。これは、微分方程式を定義した際に、従属変数の配列 yy[] の先頭 (0 番) の値を、nstep 回追うものです。ここまでの例では、0 番には振動子の座標が入っていました。結果は、独立変数 (時刻) の値と従属変数の値を Point2D.Double クラスのインスタンスとして生成し、そのリストを戻します。

このメソッドの拡張として

```
public List<Point2D.Double> evolution(double h, int nstep,
                                      int xNum, int yNum)
```

があります。二つの従属変数の xNum 番と yNum 番の組を Point2D.Double クラスのインスタンスとして生成し、そのリストを戻すものです。

課題 8　前述のメソッドを利用することで、変位を x 軸に、速度を y 軸に出力し、様子を観察しなさい。変位は従属変数の 0 番、速度は 1 番として定義されていることに留意しなさい。

HarmonicOscillator.java

```
package oscillators;

import java.awt.geom.Point2D;
import java.io.BufferedWriter;
import java.io.IOException;
import java.util.List;
import myLib.rungeKutta.Dynamics;
import myLib.utils.FileIO;

/**
 * 調和振動子
 *
 * @author tadaki
 */
public class HarmonicOscillator extends Dynamics {

    private final double omega;//角速度

    /**
     * コンストラクタ
     *
     * @param x 振幅の初期値
     * @param v 速度の初期値
     * @param k k/mに相当
     */
    public HarmonicOscillator(double x, double v, double k) {
        super(x, v);//スーパークラスによる初期化
        this.omega = Math.sqrt(k);
        //微分方程式の記述
        equation = (double xx, double[] yy) -> {

                        ここを記述


        };
    }

    /**
     * @param args the command line arguments
     * @throws java.io.IOException
     */
    public static void main(String[] args) throws IOException {
        double x0 = 0.;
        double v0 = 1.;
        double k = 1.;// k/mに相当
        HarmonicOscillator sys = new HarmonicOscillator(x0,v0,k);
```

```
        double t = 50.;
        int nstep = 10000;
        // 時間50を10000に区分して、積分を実行
        // 結果を(t,x)のリストで得る
        List<Point2D.Double> points = sys.evolution(t, nstep);
        // 結果をファイルへ出力
        BufferedWriter out = FileIO.openWriter("output.txt");
        for (Point2D.Double p : points) {
            FileIO.writeSSV(out, p.x, p.y);
        }
    }

}
```