5. 基本的な力学問題

2020/11/2

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。

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

この方法では、$O(h^4)$ までの精度で数値計算を行うことができます。

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。調和振動子を記述する運動方程式は、その位置 x に対する二階微分方程式です。

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

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

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

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

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

です。

今回は、プロジェクト DifferentialEquation のパッケージ oscillators を使います。このパッケージでは、調和振動子のほか、摩擦のある振動子や非線形振動子のクラスを含みます。これらは、抽象クラス AbstractOscillator の派生となっています。各振動子は、AbstractOscillator クラスに具体的運動方程式を追記することで定義します。

AbstractOscillator クラスの evolution() メソッドを、ソースコード 2.1 に示します。前回の Euler 法と同様に、時間を tmax だけ進めるために、時間ステップを nstep 個に区切っていきます。戻り値に注目してください。OscillatorState クラスは、時刻、変位、速度を組にして保持するだけのクラスです。Runge-Kutta 法によって、時間を進め、各時刻の状態を OscillatorState クラスに保存し、そのリストを返します。

ソースコード 2.1　時間を nstep 個進めるメソッド

```
1     public List<OscillatorState> evolution(double tmax, int nstep) {
2         double dt = tmax / nstep;
3         List<OscillatorState> timeSequence = Utils.createList();
4         for (int i = 0; i < nstep; i++) {
5             update(dt);
6             timeSequence.add(new OscillatorState(t, y[0], y[1]));
7         }
8         return timeSequence;
9     }
```

ソースコード 2.2　OscillatorState クラス

```
1  public class OscillatorState {
2      public final double t;
3      public final double x;
4      public final double v;
5
6      public OscillatorState(double t, double x, double v) {
7          this.t = t;
8          this.x = x;
9          this.v = v;
10     }
11
12 }
```

OscillatorState クラスは非常に簡単なクラスで、コンストラクタの引数を final、つまり変更不能な形で保持しているだけです (ソースコード 2.2)。各フィールドは public としていますから、直接読み出すことができます。

課題 1　クラス HarmonicOscillator を、AbstractOscillator クラスの拡張として定義します。クラスファイル HarmonicOscillator.java 中の、コンストラクタに対応する微分方程式を記載しなさい。配列 yy のうち、yy[0] が変位 x に、yy[1] が速度 v に対応していることに注意しなさい。

課題 2　HarmonicOscillator クラスの main() では、$m = 1$、$k = 1$ に対して、初期条件 $x(0) = 0$、$v(0) = 1$ を設定しています。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、ソースコード 2.3 を使ってプロットし、結果を確かめなさい。9 行目の末尾にある”\”(“¥” と同じ) は、次の行まで命令が続いていることを表しています。

ソースコード 2.3　HarmonicOscillator.plt

```
1  set terminal png fontscale 1.2
2  set xlabel "{/:Italic␣t}"
3  set ylabel "{/:Italic␣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  plot "HarmonicOscillator-output.txt" ps 2 title "simulation",\
10       sin(x) lw 3 title "sin({/:Italic␣t})"
```

課題 3　`main()` を見ると、課題 2 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、ソースファイル **??**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 4　`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。パラメタとしては、$m = 1$、$k = 1$、$\gamma = 0.1$ とし、初期条件 $x(0) = 0$、$v(0) = 1$ の下で実行しなさい。

講義で説明したように、減衰振動の運動方程式 (3.1) の一般解と、その時間微分は

$$x\,(t) = \left(A\cos\left(\omega t\right) + B\sin\left(\omega t\right)\right)e^{-\alpha t} \tag{3.2}$$

$$\dot{x}\,(t) = \omega\left(A\sin\left(\omega t\right) - B\cos\left(t\right)\right)e^{-\alpha t} - \alpha x\left(\omega t\right) \tag{3.3}$$

$$\tag{3.4}$$

です。ここで、$\alpha = \gamma/(2m)$、$\omega^2 = \left(k - \gamma^2/(4m)\right)/m$ です。初期条件 $x(0) = 0$、$v(0) = 1$ より

$$x\,(0) = A = 0 \tag{3.5}$$

$$\dot{x}\,(0) = \omega B = 1 \tag{3.6}$$

となり、解

$$x\,(t) = \frac{1}{\omega}\sin\left(\omega t\right)e^{-\alpha t} \tag{3.7}$$

を得ます。

課題 5　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**??**を参考に作成しなさい。画像の出力先ファイル名は DumpedOscillator.pdf としなさい。

課題 6　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 $\pm e^{-\alpha t}$ も描画しなさい。

3.1　発展問題：(x, v) 空間での運動

時間に余裕のある場合には、以下も実行してみましょう。

evolution() メソッドの戻り値は、各時刻での OscillatorState のリストでした。OscillatorState クラスは、時刻 t、変位 x と速度 v の組です。これを用いて、(x, v) 空間での運動を可視化しましょう。

課題 7　前述のメソッドを利用することで、減衰振動の場合に、変位を x 軸に、速度を y 軸に出力し、様子を観察しなさい。