数値シミュレーション入門
―振り子の運動を予測しよう―

第１部：振り子の運動の理論

１　ニュートン力学

われわれの身のまわりで見られる物体の運動は、通常は古典力学（ニュートン力学）によって説明される。ニュートン力学は以下の３つの法則からなる。

ニュートン力学の第１法則（慣性の法則）:
外部から力を加えられない限り、静止している物体は静止し続け、運動している物体は等速直線運動を続ける。

ニュートン力学の第２法則（運動方程式）:
物体が力を受けると力と同じ方向に加速度が生じる。加速度の大きさは力の大きさに比例し、物体の質量に反比例する。

ニュートン力学の第３法則（作用反作用の法則）:
一方の物体が他方の物体に及ぼす力と、その物体が他方の物体から受ける力は、向きが反対で大きさが等しい。

以上の３つの法則のうち、ニュートン力学の第２法則は、運動方程式によって次のように記述される。

$$ma = F$$

ただし、m は質量、a は加速度、F は力である。ここで、物体の位置 x 、速度 u 、加速度 a の間は次のような関係がある。まず、位置 x の時間微分が速度 u であって、

$$u = \frac{dx}{dt}$$

が成り立つ。また、速度 u の時間微分が加速度 a であって、

$$a = \frac{du}{dt} = \frac{d^2x}{dt^2}$$

が成り立つ。したがって、運動方程式を

$$m\frac{d^2x}{dt^2} = F$$

と書くこともできる。

２　振り子の運動の基礎

前節では運動方程式を

$$m\frac{d^2x}{dt^2}=F$$

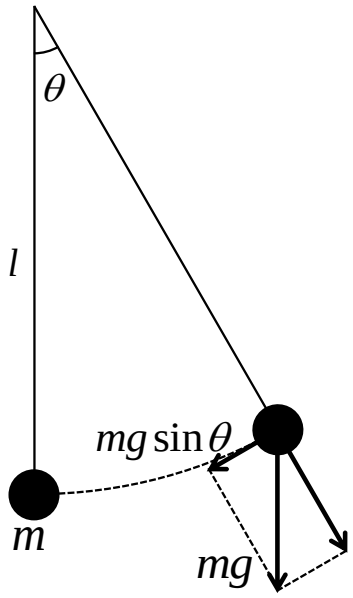
と書いた。振り子の振れ角をθとし、振り子が振れる方向にx軸を定義すると、運動方程式は、

$$m\frac{d^2x}{dt^2}=-mg\sin\theta=-mg\sin\frac{x}{l} \qquad ①$$

となって、

$$\frac{d^2x}{dt^2}=-g\sin\frac{x}{l}$$

と表せる。lは振り子の長さ、gは重力加速度を表し、$g=9.8$［m/s²］である。



振り子の振れ角θが小さいときには

$$\sin\theta\cong\theta$$

と近似できるから、運動方程式を

$$\frac{d^2x}{dt^2}=-g\frac{x}{l}=-\frac{g}{l}x \qquad ①'$$

と書くこともできる。①’は関数$x(t)$を２回微分すると、もとの関数の負の定数倍（$-g/l$倍）になることを示している。このような条件を満たす関数は三角関数である。たとえば、

$$x=\sin t$$

であれば、

$$x'=\cos t$$

$$x''=-\sin t$$

であり、

$$x=\cos t$$

であれば、

$$x' = -\sin t$$
$$x'' = -\cos t$$

である。また、

$$x = \sin at$$

であれば、

$$x' = a\cos at$$
$$x'' = -a^2 \sin at$$

となる。ここで、①’に$x = \cos \omega t$を代入すると、

$$\text{（左辺）} = -\omega^2 \cos \omega t$$
$$\text{（右辺）} = -\frac{g}{l}\cos \omega t$$

となる。両辺が等しくなるためには、

$$-\omega^2 = -\frac{g}{l}$$

つまり

$$\omega = \sqrt{\frac{g}{l}}$$

でなければならない。したがって、

$$x = \cos\sqrt{\frac{g}{l}}t \qquad ②$$

である。

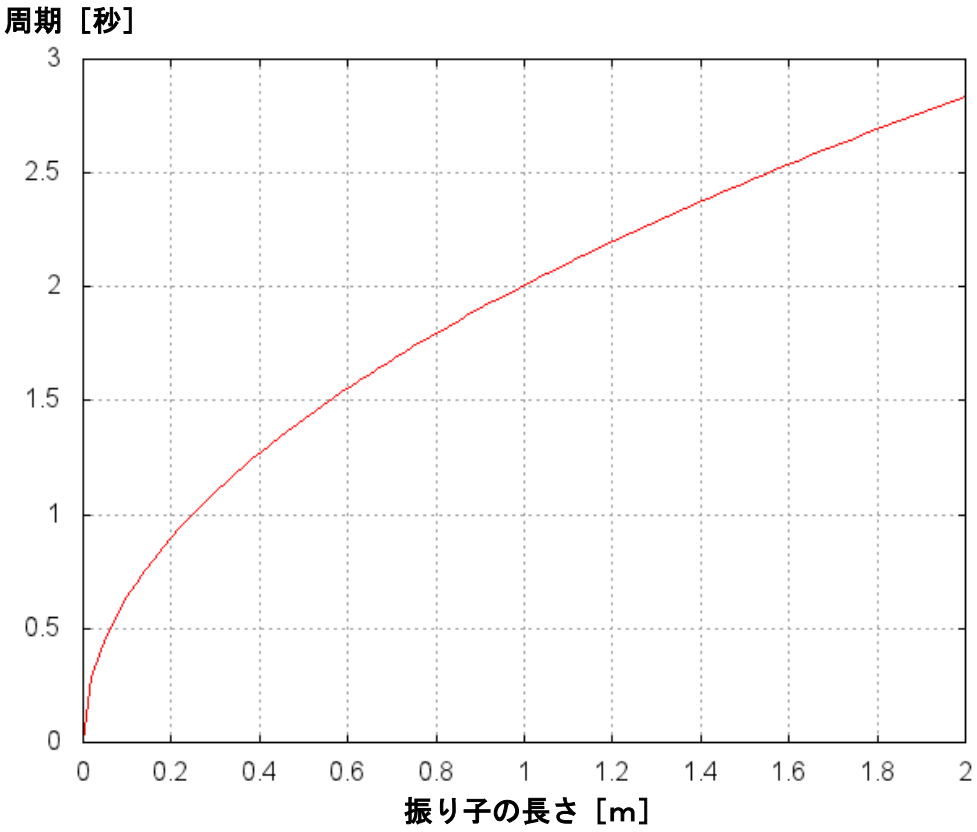
②において、$t = 0$のとき、$\sqrt{\frac{g}{l}}t = 0$である。時間tが$2\pi\sqrt{\frac{l}{g}}$だけ進むと、位相が 1 周して$\sqrt{\frac{g}{l}}t = 2\pi$となり、もとの値に戻る。つまり、②の周期は

$$T = 2\pi\sqrt{\frac{l}{g}}$$

である。周期Tを用いて、②を

$$x = \cos\frac{2\pi}{T}t$$

と書くこともできる。



振り子の長さと周期の関係

３　数値シミュレーションの方法

（１）オイラー法

数値シミュレーションによって振り子の運動を計算してみよう。速度の定義より、

$$\frac{x^+ - x}{\Delta t} = u \qquad ③$$

となるので、

$$x^+ = x + u\Delta t$$

である。ただし、x^+ は時間 Δt だけ後の x の値である。この関係式を使うと、ある時刻の x 、u の値から時間 Δt だけ後の x を計算することができる。

同様に、加速度の定義より、

$$\frac{u^+ - u}{\Delta t} = a_x \qquad ④$$

である。$a_x = -g\sin\frac{x}{l}$ を代入すると、

$$\frac{u^+ - u}{\Delta t} = -g\sin\frac{x}{l}$$

となる。したがって、

$$u^{+} = u - \left(g \sin \frac{x}{l} \right) \Delta t$$

が得られる。この関係を用いると、ある時刻の u の値から時間 Δt だけ後の u を計算することができる。このようにして次の時刻の物理量を順に計算していく手法をオイラー法という。

（２）リープフロッグ法

オイラー法で計算した振り子の運動をみると時間とともに振幅が増大しており、現実の振り子の運動とは整合しない結果になっている。この原因は、時間発展の計算式が非対称だからである。③を対称的な形に書き替えると、

$$\frac{x^{+} - x^{-}}{2\Delta t} = u \qquad ③'$$

となるので、

$$x^{+} = x^{-} + 2u\Delta t$$

である。ただし、x^{-} は時間 Δt だけ前の x の値である。この関係式を使うと、ある時刻とその時刻から Δt だけ前の時刻の x 、u の値から、Δt だけ後の x を計算することができる。

同様に、④は

$$\frac{u^{+} - u^{-}}{2\Delta t} = a_x \qquad ④'$$

と書き替えることができ、$a_x = -g \sin \frac{x}{l}$ を代入すると、

$$\frac{u^{+} - u^{-}}{2\Delta t} = -g \sin \frac{x}{l}$$

となる。したがって、

$$u^{+} = u^{-} - 2\left(g \sin \frac{x}{l} \right) \Delta t$$

が得られ、ある時刻とその時刻から Δt だけ前の時刻の x 、u の値から、Δt だけ後の u を計算することができる。このようにして次の時刻の物理量を順に計算していく手法をリープフロッグ法という。リープフロッグ法を用いると、振り子の振幅は一定に保たれ、振り子の運動を適切にシミュレーションすることができる。

第２部　振り子の運動の実験

シミュレーションの前に実験してみよう

１０往復する時間を何回かはかり、１往復あたりの時間を計算する。

実験１：ふりこの長さを変える

同じにする条件：おもりの重さ（　　）、ふれはば（　　）

ふりこの長さ	１回め	２回め	３回め	合計	１０往復する時間	１往復する時間

実験２：ふれはばを変える

同じにする条件：おもりの重さ（　　）、ふりこの長さ（　　）

ふれはば	１回め	２回め	３回め	合計	１０往復する時間	１往復する時間

【参考】小学校学習指導要領解説　理科編

第３節　第５学年

２　内容

Ａ　物質・エネルギー

(2)　振り子の運動

おもりを使い，おもりの重さや糸の長さなどを変えて振り子の動く様子を調べ，振り子の運動の規則性についての考えをもつことができるようにする。 ア　糸につるしたおもりが１往復する時間は，おもりの重さなどによっては変わらないが，糸の長さによって変わること。

本内容は，第３学年「Ａ(2)風やゴムの働き」の学習を踏まえて，「エネルギー」についての基本的な見方や概念を柱とした内容のうちの「エネルギーの見方」にかかわるものである。

ここでは，振り子の運動の規則性について興味・関心をもって追究する活動を通して，振り子の運動の規則性について条件を制御して調べる能力を育てるとともに，それらについての理解を図り，振り子の運動の規則性についての見方や考え方をもつことができるようにすることがねらいである。

ア　振り子の運動の変化に関係する条件として，児童が想定するものとしては，おもりの重さ，糸の長さ，振れ幅が考えられる。ここでは，糸におもりをつるし，おもりの重さ，または糸の長さを変えながら，おもりの１往復する時間を測定する。おもりの重さを変えて調べるときには，糸の長さやおもりの振れ幅など他の条件は一定にして調べる必要がある。それらの測定結果から，糸につるしたおもりの１往復する時間は，おもりの重さなどによっては変わらないが，糸の長さによって変わることをとらえるようにする。

ここでの指導に当たっては，糸の長さや振れ幅を一定にしておもりの重さを変えるなど，変える条件と変えない条件を制御して実験を行うことによって，実験結果を適切に処理し，考察することができるようにする。その際，**適切な振れ幅で実験を行い，振れ幅が極端に大きくならないようにする**。また，伸びの少ない糸を用い，糸の長さは糸をつるした位置からおもりの重心までであることに留意する。さらに，実験を複数回行い，その結果を処理する際には，算数科の学習と関連付けて適切に処理するようにする。

なぜだろう？

第３部：振り子の運動のシミュレーション

プログラミング作業の流れ

まずターミナルを起動する。　（⇒簡単操作マニュアル１）

☞ターミナル（端末）がすべての作業の起点になります。

１．Emacs でプログラムを書く。　（⇒１－②、２－②）

２．コンパイルする。　（⇒１－③）

３．実行する。　（⇒１－④）

４．結果を確かめる。　（⇒１－⑤⑥）

簡単操作マニュアル（Ｗｉｎｄｏｗｓ用）

１．ターミナルの使い方

ターミナルを起動する	デスクトップの「 msys.bat へのショートカット 」を ダブルクリック
①ファイルの一覧を表示する	$ ls ↵
ファイルをコピーする	$ cp *コピー元ファイル名* *コピー先ファイル名* ↵
ファイル名を変更する	$ mv *変更前のファイル名* *変更後のファイル名* ↵
ファイルを消去する	$ rm *ファイル名* ↵
②Ｅｍａｃｓを起動する	$ emacs *ファイル名* & ↵ （または emacs & ↵）
③コンパイルする	$ cc *ファイル名* ↵ math.h を使う場合： $ cc *ファイル名* -lm ↵
④実行する	$ ./a.exe ↵
⑤ファイルの中身を見る	$ less *ファイル名* ↵ 矢印キーで移動、Ｑを押して終了
⑥ｇｎｕｐｌｏｔを起動する	$ gnuplot ↵
ターミナルを終了する	$ exit ↵

２．Ｅｍａｃｓの使い方

①ファイルを開く	コントロールキー＋Ｘ コントロールキー＋Ｆ →ファイル名を入力
②ファイルを保存する	コントロールキー＋Ｘ コントロールキー＋Ｓ
※操作の取り消し	コントロールキー＋Ｇ
Ｅｍａｃｓを終了する	コントロールキー＋Ｘ コントロールキー＋Ｃ

３．ｇｎｕｐｌｏｔの使い方

①グラフをかく	> plot ”*ファイル名*” ↵ 複数の場合： plot ”*ファイル名*”， ”*ファイル名*”， … ↵
②折れ線グラフにする	> set style data lines ↵
範囲を指定する	> set xrange [*0*:*50*] ↵ > set yrange [*0*:*30*] ↵
③再描画する	> replot ↵
※画像ファイルに保存する	画面上で作図したあとで： > set term png ↵ > set output ”*ファイル名.png*” ↵ > replot ↵
ｇｎｕｐｌｏｔを終了する	> quit ↵

簡単操作マニュアル（Ｌｉｎｕｘ用）

１．ターミナルの使い方

ターミナルを起動する	ランチャー（画面左側）の「端末」をクリック
①ファイルの一覧を表示する	> ls
ファイルをコピーする	> cp *コピー元ファイル名* *コピー先ファイル名*
ファイル名を変更する	> mv *変更前のファイル名* *変更後のファイル名*
ファイルを消去する	> rm *ファイル名*
②Ｅｍａｃｓを起動する	> emacs & または emacs *ファイル名* &
③コンパイルする	> cc *ファイル名* 必要に応じて(math.h を使う場合)： cc *ファイル名* -lm
④実行する	> ./a.out または ./a.exe
⑤ファイルの中身を見る	> less *ファイル名* 矢印キーで移動、Ｑを押して終了
⑥ｇｎｕｐｌｏｔを起動する	> gnuplot
ターミナルを終了する	> exit

２．Ｅｍａｃｓの使い方

①ファイルを開く	コントロールキー＋Ｘ　コントロールキー＋Ｆ →ファイル名を入力
②ファイルを保存する	コントロールキー＋Ｘ　コントロールキー＋Ｓ
※操作の取り消し	コントロールキー＋Ｘ　コントロールキー＋Ｇ
Ｅｍａｃｓを終了する	コントロールキー＋Ｘ　コントロールキー＋Ｃ

３．ｇｎｕｐｌｏｔの使い方

①グラフをかく	> plot ”*ファイル名*” 複数の場合： plot ”*ファイル名*”, ”*ファイル名*”, …
②折れ線グラフにする	> set style data lines
範囲を指定する	> set xrange [*0*:*50*] > set yrange [*0*:*30*]
③再描画する	> replot
※画像ファイルに保存する	画面上で作図したあとで： > set term png > set output ”*ファイル名.png*” > replot
ｇｎｕｐｌｏｔを終了する	> quit

プログラムの基本構造

```
#include  <stdio.h>
#include  <math.h>     /*  この行は必要な場合※のみ  */
int  main  (void)
{

/*  ここに処理の内容を書く  */

    return  0;
}
```

※平方根や三角関数などを使う場合

①周期を計算しよう

作業の手順：以下のサンプルプログラムを Emacs で作成しよう。

☞プログラムのファイル名は自由だが、「.c」で終わる必要がある。
　例：prog01.c なら　　⇒　　ターミナル上で　$ emacs　prog01.c　&↵
☞書き終わったら忘れずに保存する（コントロールキー＋Ｘ　コントロールキー＋Ｓ）。
コントロールキーを押しながらＸを押す

```
#include  <stdio.h>
#include  <math.h>
int  main  (void)
{

    float  g,  L,  T;                                   /*  変数を宣言する  */

    g  =  9.8;                                          /*  定数を代入する  */

    printf  ("Length  [m]?\n");
    scanf  ("%f",  &L);                                 /*  条件を入力する  */

    T  =  2.0  *  3.14159  *  sqrt (L  /  g);

    printf  ("T  =  %9.3f\n",  T);                      /*  結果を出力する  */

    return  0;
}
```

メモ：

$$T = 2\pi\sqrt{\frac{l}{g}}$$

実行例：ターミナル上でコンパイルして実行してみよう。

$ cc prog01.c -lm ↵	コンパイルします。
$ ls ↵	実行ファイルを確認します。
a.exe prog01.c	☞環境によってはa.exeではなくa.outになる場合もあります。
$ ./a.exe ↵	実行します。
Length [m]?	
1 ↵	数値を入力します。
T = 2.007	結果が出力されます。

メモ：

$$L = 1\,[\mathrm{m}] \quad \Rightarrow \quad T \fallingdotseq 2.0\,[\mathrm{s}]$$

☞正常に実行できることを確認したら、Emacsを終了してよい
（コントロールキー＋Ｘ　コントロールキー＋Ｃ）。

②初期の状態を計算しよう

作業の手順：①で作成したプログラムをコピーして、Emacs で完成させよう。

☞コピーするときには cp コマンドを使う。
　例：$ cp　prog01.c　prog02.c ↵　　（⇒簡単操作マニュアル１－①）
☞コピーしたファイルを改めて emacs で開く。

```
#include  <stdio.h>
#include  <math.h>
int  main  (void)
{

    float  g,  L,  angle,  theta,  x,  u;          /*  変数を宣言する  */

    g  =  9.8;                                     /*  定数を代入する  */

    printf  ("Length  [m],  angle  [deg.]?\n");
    scanf  ("%f,%f",  &L,  &angle);                /*  条件を入力する  */
    theta  =  3.14159  /  180.0  *  angle;

    x  =  L  *  theta;                             /*  初期値を計算する  */
    u  =  0.0;                                       ここで初期値を計算します。

    printf  ("%9.3f  %9.3f\n",  0.0,  angle);      /*  結果を出力する  */

    return  0;
}
```

実行例：ターミナル上でコンパイルして実行してみよう。

```
$  cc  prog02.c  -lm ↵          コンパイルします。

$ ./a.exe ↵                    実行します。

Length  [m],  angle  [deg.]?

1,15 ↵                         数値を入力します。

     0.000      15.000         結果が出力されます。
```

③0.01 秒後の状態を計算しよう

作業の手順：②で作成したプログラムをコピーして、Emacs で完成させよう。

```
#include  <stdio.h>
#include  <math.h>
int  main  (void)
{

    float  g,  L,  angle,  theta,  x,  u,
           dxdt,  dudt,  t;                          /*  変数を宣言する  */

    g  =  9.8;                                       /*  定数を代入する  */

    printf  ("Length  [m/s],  angle  [deg.]?\n");
    scanf  ("%f,%f",  &L,  &angle);                  /*  条件を入力する  */
    theta  =  3.14159  /  180.0  *  angle;

    x  =  L  *  theta;                               /*  初期値を計算する  */
    u  =  0.0;

    printf  ("%9.3f  %9.3f\n",  0.0,  angle);        /*  結果を出力する  */

    dxdt   =  u;                                     /*  時間微分を計算する  */
    theta  =  x  /  L;                               /*  角度を計算する  */
    dudt   =  -  g  *  sin  (theta);                 /*  時間微分を計算する  */

    x  =  x  +  0.01  *  dxdt;                       /*  次の時刻の値を計算する  */
    u  =  u  +  0.01  *  dudt;                           ここで 0.01 秒後の値を
                                                         計算します。

    t  =  0.01;                                      /*  経過時間を計算する  */
    theta  =  x  /  L;                               /*  角度を計算する  */
    angle  =  180.0  /  3.14159  *  theta;

    printf  ("%9.3f  %9.3f\n",  t,  angle);          /*  結果を出力する  */

    return  0;
}
```

実行例：ターミナル上でコンパイルして実行してみよう。

```
$  cc  prog03.c  -lm ↵
$ ./a.exe ↵
Length  [m],  angle  [deg.]?

1,15 ↵
     0.000      15.000
     0.010      15.000
```

コンパイルします。

実行します。

数値を入力します。

結果が出力されます。

④結果をファイルに書き出そう

作業の手順：③で作成したプログラムをコピーして、Emacs で完成させよう。

実行したら出力ファイルの内容を確認しよう。

```
#include  <stdio.h>
#include  <math.h>
int  main  (void)
{

    float  g,  L,  angle,  theta,  x,  u,
           dxdt,  dudt,  t;                          /*  変数を宣言する  */
    FILE  *fp;


    g  =  9.8;                                       /*  定数を代入する  */


    printf  ("Length  [m],  angle  [deg.]?\n");
    scanf  ("%f,%f",  &L,  &angle);                  /*  条件を入力する  */
    theta  =  3.14159  /  180.0  *  angle;

    x  =  L  *  theta;                               /*  初期値を計算する  */
    u  =  0.0;


    fp  =  fopen  ("output.txt",  "w");              /*  ファイルを開く  */
    fprintf  (fp,  "%9.3f  %9.3f\n",  0.0,  angle);
                                                     /*  結果をファイルに出力する  */


    dxdt   =  u;                                     /*  時間微分を計算する  */
    theta  =  x  /  L;                               /*  角度を計算する  */
    dudt   =  -  g  *  sin  (theta)                  /*  時間微分を計算する  */;

    x  =  x  +  0.01  *  dxdt;                       /*  次の時刻の値を計算する  */
    u  =  u  +  0.01  *  dudt;


    t  =  0.01;                                      /*  経過時間を計算する  */
    theta  =  x  /  L;                               /*  角度を計算する  */
    angle  =  180.0  /  3.14159  *  theta;


    fprintf  (fp,  "%9.3f  %9.3f\n",  t,  angle);
                                                     /*  結果をファイルに出力する  */


    fclose  (fp);                                    /*  ファイルを閉じる  */
    return  0;
}
```

実行例：ターミナル上でコンパイルして実行してみよう。

```
$  cc  prog04.c  -lm ↵        コンパイルします。

$ ./a.exe ↵                  実行します。

Length  [m],  angle  [deg.]?

1,15 ↵                       数値を入力します。

$  ls ↵                      出力ファイルを確認します。

a.exe  output.txt  prog01.c  prog02.c  prog03.c  prog04.c

$  less  output.txt ↵        出力ファイルの中身を見ます。→見終わったらＱで終了。
```

⑤とりあえず 10 秒後まで計算しよう

作業の手順：④で作成したプログラムをコピーして、Emacs で完成させよう。

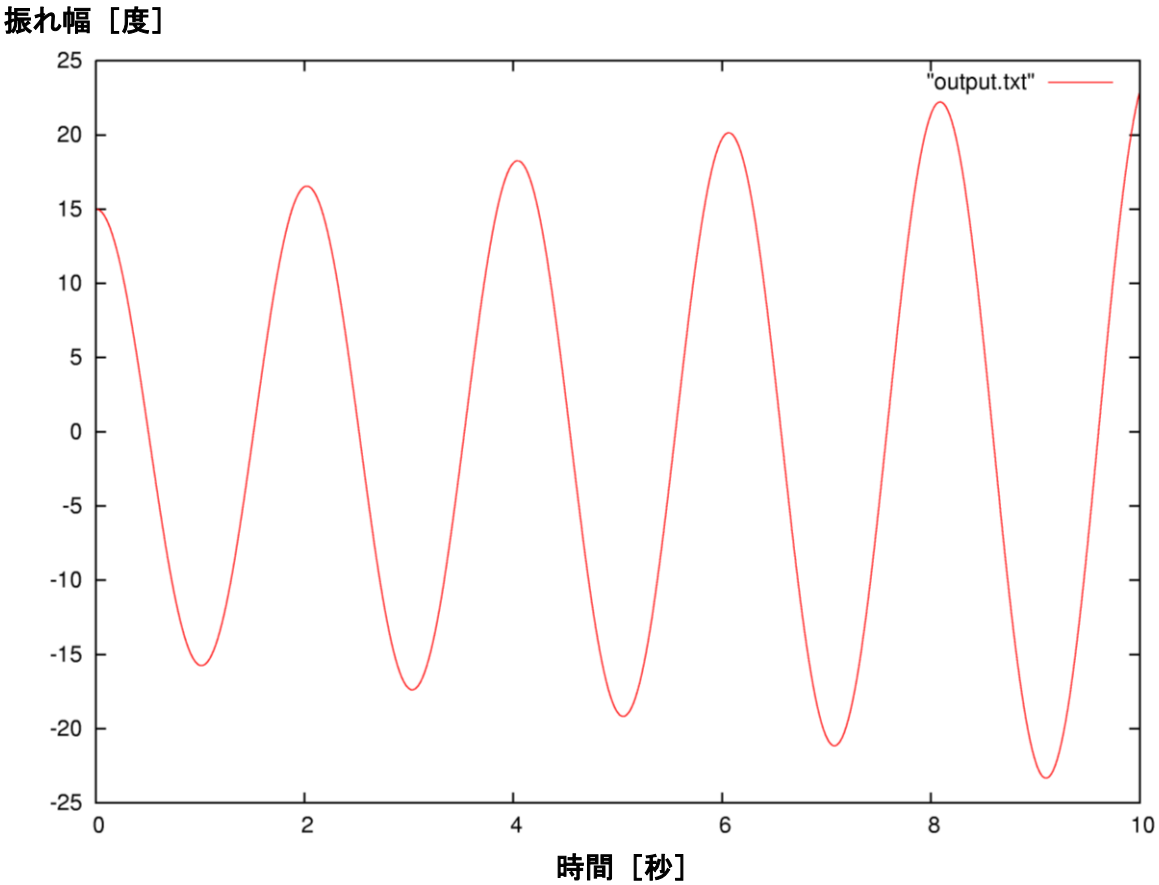
実行したら出力ファイルの内容を gnuplot で作図しよう。

```
#include  <stdio.h>
#include  <math.h>
int  main  (void)
{


    int  i;                                          ☞int(整数)と float(実数)を区別します。
    float  g,  L,  angle,  theta,  x,  u,
           dxdt,  dudt,  t;                          /*  変数を宣言する  */
    FILE  *fp;


    g  =  9.8;                                       /*  定数を代入する  */


    printf  ("Length  [m],  angle  [deg.]?\n");
    scanf  ("%f,%f",  &L,  &angle);                  /*  条件を入力する  */
    theta  =  3.14159  /  180.0  *  angle;


    x  =  L  *  theta;                               /*  初期値を計算する  */
    u  =  0.0;


    fp  =  fopen  ("output.txt",  "w");              /*  ファイルを開く  */
    fprintf  (fp,  "%9.3f  %9.3f\n",  0.0,  angle);
                                                     /*  結果をファイルに出力する  */


    for  (i=1;  i<=1000;  i++)                       /*  同じ処理を 1000 回繰り返す  */
    {


        dxdt  =  u;                                  /*  時間微分を計算する  */
        theta  =  x  /  L;                           /*  角度を計算する  */
        dudt  =  -  g  *  sin  (theta);              /*  時間微分を計算する  */


        x  =  x  +  0.01  *  dxdt;                   /*  次の時刻の値を計算する  */
        u  =  u  +  0.01  *  dudt;


        t  =  0.01  *  i;                            /*  経過時間を計算する  */
        theta  =  x  /  L;                           /*  角度を計算する  */
        angle  =  180.0  /  3.14159  *  theta;


        fprintf  (fp,  "%9.3f  %9.3f\n",  t,  angle);
                                                     /* 結果をファイルに出力する */


    }


    fclose  (fp);                                    /*  ファイルを閉じる  */
    return  0;
```

```
}
```

作図例：実行した後、less で結果を確認したら、gnuplot で作図しよう。

$ gnuplot ↵	gnuplot を起動します。
gnuplot> set style data lines ↵	折れ線グラフを指定します。
gnuplot> plot ”output.txt” ↵	グラフをかきます。
gnuplot> set xrange [0:10] ↵	範囲を指定します。
gnuplot> set yrange [-25:25] ↵	
gnuplot> replot ↵	再描画します。



計算結果の例（長さ１ｍ、振幅 15° の場合）

→ 時間とともに振幅が大きくなる。

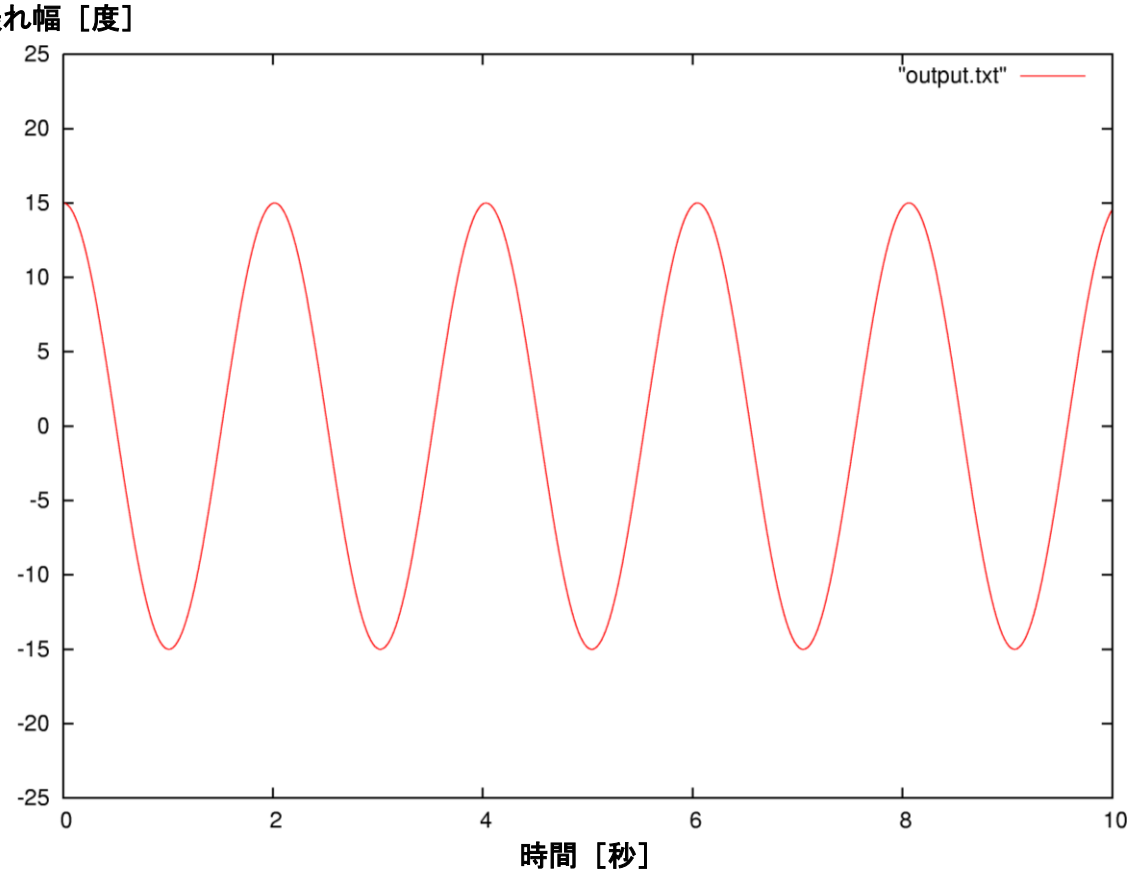
☞結果を確認したら、quit とタイプして、gnuplot を終了する。

⑥リープフロッグ法で計算しよう

作業の手順：⑤で作成したプログラムをコピーして、Emacs で完成させよう。
実行したら出力ファイルの内容を gnuplot で作図しよう。

```
#include  <stdio.h>
#include  <math.h>
int  main  (void)
{

    int  i;
    float  g,  L,  angle,  theta,  x,  u,
           xminus,  uminus,  xplus,  uplus,
           dxdt,  dudt,  t;                          /*  変数を宣言する  */
    FILE  *fp;

    g  =  9.8;                                       /*  定数を代入する  */

    printf  ("Length  [m],  angle  [deg.]?\n");
    scanf  ("%f,%f",  &L,  &angle);                  /*  条件を入力する  */
    theta  =  3.14159  /  180.0  *  angle;

    x  =  L  *  theta;                               /*  初期値を計算する  */
    u  =  0.0;

    fp  =  fopen  ("output.txt",  "w");              /*  ファイルを開く  */
    fprintf  (fp,  "%9.3f  %9.3f\n",  0.0,  angle);
                                                     /*  結果をファイルに出力する  */

    for  (i=1;  i<=1000;  i++)                       /*  同じ処理を 1000 回繰り返す  */
    {

        dxdt   =  u;                                 /*  時間微分を計算する  */
        theta  =  x  /  L;                           /*  角度を計算する  */
        dudt   =  -  g  *  sin  (theta);             /*  時間微分を計算する  */

        if (i == 1)                                  /*  次の時刻の値を計算する  */
        {

            xplus  =  x  +  0.01  *  dxdt;           /*  初回だけ計算方法を変える  */
            uplus  =  u  +  0.01  *  dudt;

        }else{

            xplus  =  xminus  +  2.  *  0.01  *  dxdt;
            uplus  =  uminus  +  2.  *  0.01  *  dudt;
```

```
        }

        xminus  =  x;                                   /*  次の時刻に進むために  */
        uminus  =  u;                                   /*    x を x^-に代入する  */

        x  =  xplus;                                    /*  次の時刻に進むために  */
        u  =  uplus;                                    /*    x^+を x に代入する  */

        t  =  0.01  *  i;                               /*  経過時間を計算する  */
        theta  =  x  /  L;                              /*  角度を計算する  */
        angle  =  180.0  /  3.14159  *  theta;

        fprintf  (fp,  "%9.3f  %9.3f\n",  t,  angle);
                                                        /* 結果をファイルに出力する */

    }

    fclose  (fp);                                       /*  ファイルを閉じる  */
    return  0;
}
```



計算結果の例（長さ１ｍ、振幅 15° の場合）

→ 振幅は一定。

☞結果を確認したら、quit とタイプして、gnuplot を終了する。

⑦振り子の長さを変えてみよう

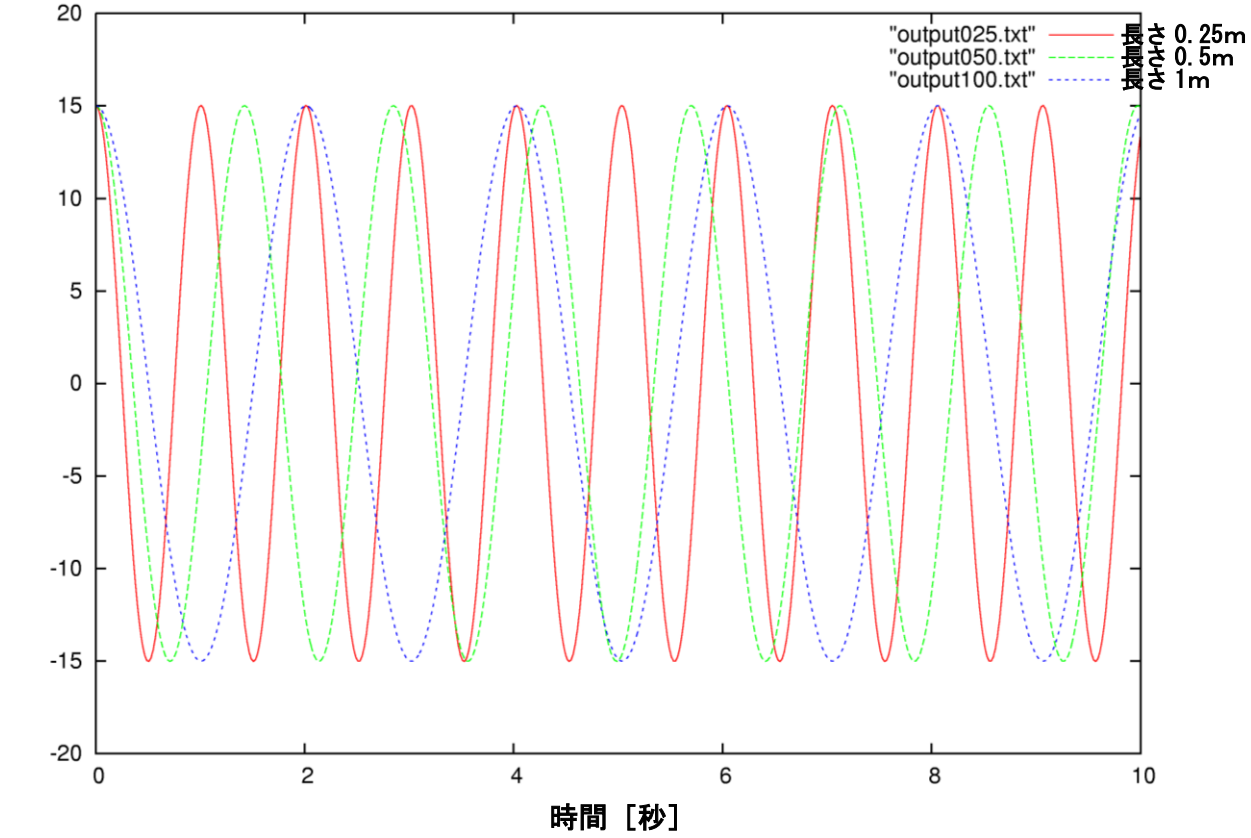
作業の手順：⑥で作成したプログラムで、振幅は一定のまま、
振り子の長さを変えて計算しよう。
計算ごとに出力ファイル名を変えて保存しよう。
出力ファイルの内容をまとめて gnuplot で作図しよう。

振り子の長さ１　ｍ、振幅 15°で実験　→　出力ファイル名を output100.txt に変更。
振り子の長さ 0.5 ｍ、振幅 15°で実験　→　出力ファイル名を output050.txt に変更。
振り子の長さ 0.25ｍ、振幅 15°で実験　→　出力ファイル名を output025.txt に変更。
☞ファイルの名前を変更するときには mv コマンドを用いる（簡単操作マニュアル参照）。

例：　$ ./a.exe ↵
　　Length　[m],　angle　[deg.]?
　　1,15 ↵
　　$ mv　output.txt　output100.txt ↵

→ gnuplot で３つの出力ファイルをまとめて作図（簡単操作マニュアル参照）。

例：　$ gnuplot ↵
　　gnuplot>　set　style　data　lines ↵
　　gnuplot>　plot　”output100.txt”,　”output050.txt”, … ↵



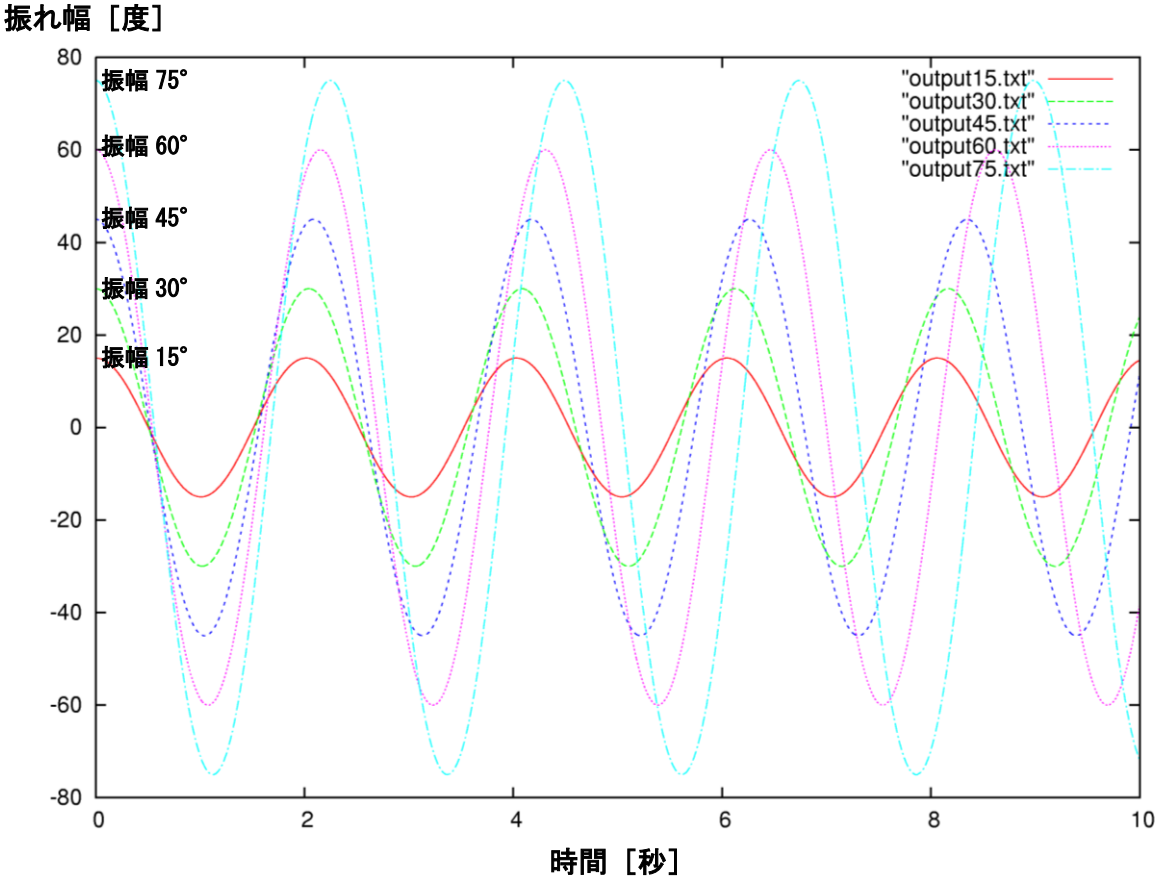
計算結果の例（振幅 15° の場合）

→ 振り子の長さが短いほうが周期も短くなる。

⑧振幅を変えてみよう

作業の手順：⑥で作成したプログラムで、振り子の長さは一定のまま、
振幅を変えて計算しよう。
計算ごとに出力ファイル名を変えて保存しよう。
出力ファイルの内容をまとめて gnuplot で作図しよう。

振り子の長さ 1m、振幅 15°で実験　→　出力ファイル名を output15.txt に変更。
振り子の長さ 1m、振幅 30°で実験　→　出力ファイル名を output30.txt に変更。
振り子の長さ 1m、振幅 45°で実験　→　出力ファイル名を output45.txt に変更。
振り子の長さ 1m、振幅 60°で実験　→　出力ファイル名を output60.txt に変更。
振り子の長さ 1m、振幅 75°で実験　→　出力ファイル名を output75.txt に変更。
☞ファイルの名前を変更するときには mv コマンドを用いる（簡単操作マニュアル参照）。
→ gnuplot で５つの出力ファイルをまとめて作図（簡単操作マニュアル参照）。



計算結果の例（振り子の長さ１ｍの場合）

→ 振幅が大きいほうが周期が長くなる。

発展：振幅と周期の関係

ここでは発展的な内容として、微小振幅ではない場合において振幅と周期の関係を数学的に計算する方向を解説します。大学レベルの数学を用いた内容であり、小中学校はもちろん、高等学校の物理においても範囲外です。難しい部分もありますので、特に興味のある方は参考にしてください。

振り子の振幅（振れ角θの最大値）を$\theta_{\max}$とする。振り子の位置（振れ角）が$\theta=\theta_{\max}$のとき、力学的エネルギーは、

$$mgl(1-\cos\theta_{\max})$$

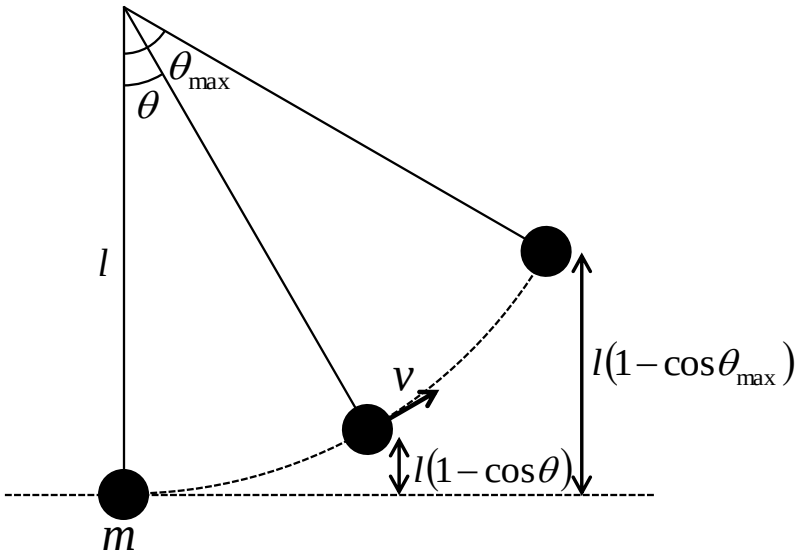
である。なぜなら、運動エネルギーはゼロであり、位置エネルギーのみを考えればよいからである。一般に、振り子の位置がθのときには、力学的エネルギーは運動エネルギーと位置エネルギーの和として、

$$\frac{1}{2}mv^2+mgl(1-\cos\theta)$$

と書ける。したがって、力学的エネルギー保存則より、

$$\frac{1}{2}mv^2+mgl(1-\cos\theta)=mgl(1-\cos\theta_{\max}) \qquad ①$$

が成り立つ。



①を変形すると、

$$\frac{1}{2}v^2=gl(\cos\theta-\cos\theta_{\max})$$

$$v=\sqrt{2gl(\cos\theta-\cos\theta_{\max})} \qquad ②$$

が得られる。ここで、振り子の位置が$\Delta\theta$だけ変化するのにかかる時間Δtを考えてみる。時間Δtの間に振り子が移動しなければならない距離は$l\Delta\theta$であり、振り子の速さはvだから、

$$\Delta t=\frac{l\Delta\theta}{v}=\sqrt{\frac{l}{2g(\cos\theta-\cos\theta_{\max})}}\Delta\theta \qquad ③$$

が成り立つ。③を微分の形で書くと、

$$\frac{dt}{d\theta}=\sqrt{\frac{l}{2g\left(\cos\theta-\cos\theta_{\max}\right)}} \qquad ④$$

である。④を$\theta=0$から$\theta=\theta_{\max}$まで積分すれば、振り子が$\theta=0$から$\theta=\theta_{\max}$まで移動するのにかかる時間が求められるはずである。これは４分の１周期に相当するから、周期Tを求めるためには、

$$T=4\int_0^{\theta_{\max}}\frac{dt}{d\theta}d\theta=4\int_0^{\theta_{\max}}\sqrt{\frac{l}{2g\left(\cos\theta-\cos\theta_{\max}\right)}}d\theta \qquad ⑤$$

を計算すればよい。三角関数に関する公式

$$\cos\theta=1-2\sin^2\frac{\theta}{2}$$

を用いると、⑤は

$$T=2\int_0^{\theta_{\max}}\sqrt{\frac{l}{g\left(\sin^2\frac{\theta_{\max}}{2}-\sin^2\frac{\theta}{2}\right)}}d\theta \qquad ⑥$$

と変形できる。ここで、

$$\sin\alpha=\frac{\sin\frac{\theta}{2}}{\sin\frac{\theta_{\max}}{2}} \qquad ⑦$$

とおくと、

$$\cos\alpha d\alpha=\frac{\cos\frac{\theta}{2}}{2\sin\frac{\theta_{\max}}{2}}d\theta$$

だから、

$$\frac{d\theta}{d\alpha}=2\cos\alpha\frac{\sin\frac{\theta_{\max}}{2}}{\cos\frac{\theta}{2}}=2\cos\alpha\sin\frac{\theta_{\max}}{2}\frac{1}{\sqrt{1-\sin^2\frac{\theta_{\max}}{2}\sin^2\alpha}} \qquad ⑧$$

である。⑦、⑧を用いて、⑥を置換積分すると、

$$T=2\int_0^{\frac{\pi}{2}}\frac{1}{\sin\frac{\theta_{\max}}{2}}\sqrt{\frac{l}{g\left(1-\sin^2\alpha\right)}}\frac{d\theta}{d\alpha}d\alpha=4\sqrt{\frac{l}{g}}\int_0^{\frac{\pi}{2}}\frac{1}{\sqrt{1-\sin^2\frac{\theta_{\max}}{2}\sin^2\alpha}}d\alpha \qquad ⑨$$

と変形できる。ここで、

$$m=\sin^2\frac{\theta_{\max}}{2}$$

とおくと、

$$T = 4\sqrt{\frac{l}{g}}\int_0^{\frac{\pi}{2}} \frac{1}{\sqrt{1-m\sin^2\alpha}} d\alpha \qquad ⑩$$

のように表すことができる。⑩をみると、微小振幅、つまり $m \to 0$ のとき、$T \to 2\pi\sqrt{\frac{l}{g}}$ であることが確かめられる。

さて、⑩に出てきた

$$K(m) = \int_0^{\frac{\pi}{2}} \frac{1}{\sqrt{1-m\sin^2\theta}} d\theta$$

の形の積分は、第１種完全楕円積分とよばれている。⑩を計算するために、⑩の中の被積分関数をテイラー展開すると、

$$\frac{1}{\sqrt{1-m\sin^2\alpha}} = 1+\frac{1}{2}m\sin^2\alpha+\frac{3}{8}m^2\sin^4\alpha+\frac{15}{48}m^3\sin^6\alpha+\cdots = \sum_{n=0}^{\infty}\frac{(2n-1)!!}{(2n)!!}m^n\sin^{2n}\alpha \qquad ⑪$$

となる。⑪は $0 \le m < 1$ の範囲で一様収束するので、各項ごとに積分することができて、

$$\int_0^{\frac{\pi}{2}} \frac{1}{\sqrt{1-m\sin^2\alpha}} d\alpha = \sum_{n=0}^{\infty}\frac{(2n-1)!!}{(2n)!!}m^n\int_0^{\frac{\pi}{2}}\sin^{2n}\alpha d\alpha \qquad ⑫$$

である。ここで、部分積分の公式より、

$$\int_0^{\frac{\pi}{2}}\cos^2\alpha\sin^{2n}\alpha d\alpha = \left[\cos^2\alpha\sin^{2n+1}\alpha\right]_0^{\frac{\pi}{2}} + \frac{1}{2n+1}\int_0^{\frac{\pi}{2}}\sin^{2(n+1)}\alpha d\alpha$$

だから、

$$\int_0^{\frac{\pi}{2}}\sin^{2(n+1)}\alpha d\alpha = \frac{2n+1}{2n+2}\int_0^{\frac{\pi}{2}}\sin^{2n}\alpha d\alpha$$

となって、

$$\int_0^{\frac{\pi}{2}}\sin^{2n}\alpha d\alpha = \frac{(2n-1)!!}{(2n)!!}\frac{\pi}{2} \qquad ⑬$$

が得られる。⑬を用いて⑫を計算すると、

$$\int_0^{\frac{\pi}{2}} \frac{1}{\sqrt{1-m\sin^2\alpha}} d\alpha = \frac{\pi}{2}\sum_{n=0}^{\infty}\left\{\frac{(2n-1)!!}{(2n)!!}\right\}^2 m^n \qquad ⑭$$

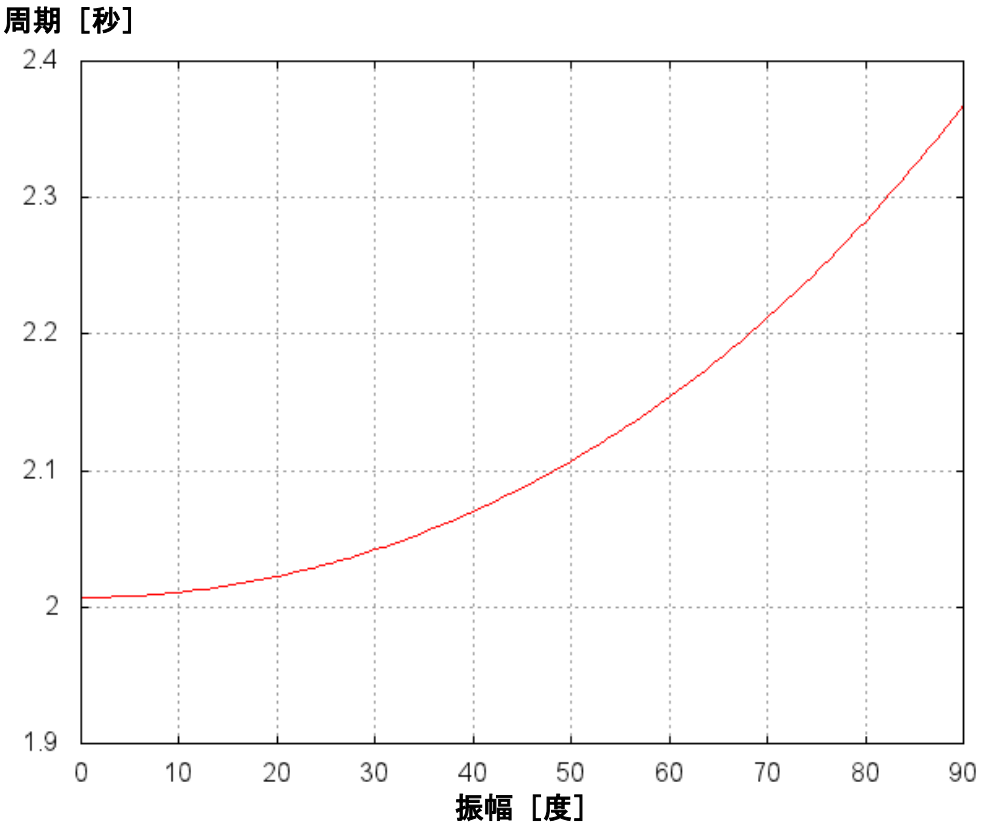
となる。⑭を⑩に代入して、

$$T = 2\pi\sqrt{\frac{l}{g}}\sum_{n=0}^{\infty}\left\{\frac{(2n-1)!!}{(2n)!!}\right\}^2 m^n = 2\pi\sqrt{\frac{l}{g}}\sum_{n=0}^{\infty}\left\{\frac{(2n-1)!!}{(2n)!!}\right\}^2 \sin^{2n}\frac{\theta_{\max}}{2} \qquad ⑮$$

と求められる。これが有限振幅の振り子の周期である。⑮の各項を書き出すと、

$$T = 2\pi\sqrt{\frac{l}{g}}\sum_{n=0}^{\infty}\left\{\frac{(2n-1)!!}{(2n)!!}\right\}^2 \sin^{2n}\frac{\theta_{\max}}{2} = 2\pi\sqrt{\frac{l}{g}}\left(1+\frac{1}{4}\sin^2\frac{\theta_{\max}}{2}+\frac{9}{64}\sin^4\frac{\theta_{\max}}{2}+\cdots\right) \qquad ⑯$$

となる。⑯を用いて周期Tを計算した結果を次の図に示す。



振幅と周期の関係（振り子の長さ１ｍの場合）

補遺：文法の解説

【変数の宣言】

プログラムの最初で、そのプログラムで用いる変数を宣言する必要がある（宣言しなければ、コンピュータにとっては意味のない文字列になってしまう)。整数型の変数の場合は int、浮動小数点型の変数（小数点以下を含む実数を表現するための変数）の場合は float で宣言する。

```
例： int  i,  j;
     float  x,  y,  z;
```

変数名の大文字と小文字は区別される。変数名は２文字以上の長さでもよい。プログラムの中では、順序や個数などを表すための整数と、連続的な数量を表すための実数（浮動小数点）は区別される。i = 0 と x = 0.0 は全く別の数である。

【メッセージを書き出す】

メッセージをターミナルに書き出すためには、printf を使う。固定されたメッセージを書き出す場合は、

```
printf ("Hello, world!\n");
```

のようにする。二重引用符で囲むことに注意。「 \n 」は改行するという意味。

【数値を書き出す】

ターミナルに数値を書き出すためには、

```
printf ("i = %9d, x = %9.3f\n", i, x);
```

のようにする。「 %9d 」や「 %9.3f 」は書式指定子とよばれる。この例では、１番目の書式指定子%9d に i の値が、２番目の書式指定子%9.3f に x の値が代入される。「 %9d 」は９桁の整数、「 %9.3f 」は９桁の実数で小数点以下は３桁であることを示している。桁数の指定を省略して、「 %d 」、「 %f 」と書いてもよい。

【数値を読み取る】

ターミナルから数値を読み取るためには、scanf を使う。たとえば、整数型の変数 i と浮動小数点型の変数 x に値を入れる場合は、

```
scanf ("%d, %f", &i, &x);
```

のようにする。この例では、コンマで区切って整数と実数を１つずつ入力し、１番目の整数が i に、２番目の実数が x に代入される。このように変数の値を取得する場合には、変数名の前に「 & 」をつける。

【処理の反復】

同じ処理を反復するためには、次のような for ループを用いて、

```
for  (i=1;  i<=1000;  i++)
{
    ..........
}
```

のようにすれば、{} 内の処理が反復する。上の例では、初め変数 i の値は 1 であり、i が 1000 以下であれば同じ処理を繰り返す。「 i++ 」は 1 回の処理が終わるごとに変数 i に 1 を加えることを示している。したがって、処理は 1000 回反復することになる。

【ファイルに書き出す】

数値をターミナルではなくファイルに書き出すためには、まず fopen で出力ファイルを開く。

```
fp = fopen ("output.txt", "w");
```

fp はファイルポインタとよばれ、開いたファイルを示す目印である。fopen ではファイル名とモードを指定する。モードは今回の場合"w"であり書き込み可能であることを示している。すでに存在するファイルを指定すると上書きされる。fopen で開いたファイルに数値を書き出すためには、fprintf を用いて、

```
fprintf (fp, "%9d %9.3f\n", i, x);
```

のようにする。基本的には printf と同じ使い方である。初めにファイルポインタを指定する点だけが異なっている。ファイルへの書き出しが終わったら、

```
fclose (fp);
```

でファイルを閉じる。なお、ファイルポインタはあらかじめ、

```
FILE *fp;
```

のように宣言しておく必要がある。

アプリケーションのインストール

※ここでは、WindowsＰＣ上にプログラミング環境を構築する方法を解説します。おもにWindows 10を想定した説明になっていますが、Windows 7やXPでも同様にインストールできます。

以下のインストール作業の解説の内容には十分に注意を払っておりますが、環境によっては指示通りに作業をしても正常にインストールできない場合があります。その場合のサポートには応じられませんので、あらかじめご了承ください。また、アプリケーションのインストールや利用の際に生じたトラブルや損害についても責任を負うことはできませんので、この点も理解のうえご利用ください。

【アカウントの準備】

ＰＣ上での自分のユーザ名（ログインするときの名前）を確認する。ユーザ名が半角英数字でない場合はインストールできない。ユーザアカウントを作成したときには半角英数字でなかったが、後で半角英数字に変更した場合も同様である。ユーザ名が半角英数字でない場合は、半角英数字のユーザ名で新しいアカウントを作成し、そのアカウントでインストールやプログラミングを行なう必要がある。

例：○ hgakugei　　× 学芸花子　　× ｈｇａｋｕｇｅｉ　（←全角英数字）

× 学芸花子からhgakugeiに変更

また、インストール作業では管理者権限が必要である。

【インストール作業】

ＣＤ－ＲＯＭの中のフォルダ「プログラミング環境」に保存されている圧縮フォルダmingw.zipをＰＣにコピーした後、すべて展開して、中身を確認する。

☞mingw.zipを右クリックして「すべて展開」を選択する。

単にダブルクリックしただけでは一時展開なので以下の作業がうまくいかない。

①フォルダMinGWをC:¥にコピーする。

☞画面左下のスタートボタンからWindowsシステムツール、エクスプローラーを選び、ローカルディスク(C:)をクリックすると、C:¥を開くことができる。

②フォルダemacs-24.3をC:¥にコピーする。

③フォルダgnuplotをC:¥にコピーする。

④システム環境変数を設定する：

スタートボタン→Windowsシステムツール→コントロールパネル

→システムとセキュリティ→システム→システムの詳細設定→詳細設定→環境変数

「システム環境変数」の中の「Path」を選択し、「編集」をクリックする。
「新規」をクリックして１行に１項目ずつ、「 C:¥MinGW¥bin 」、「 C:¥emacs-24.3¥bin 」、
「 C:¥gnuplot¥bin 」の合計３項目を追加する。
※Windows 7、XP では、「編集」をクリックした後、「 ; 」で区切りながら、
「 C:¥MinGW¥bin;C:¥emacs-24.3¥bin;C:¥gnuplot¥bin 」を追加する。
☞すでに書かれている内容の末尾にセミコロンを付け加え、その後にかぎ括弧の内容を追記する。

注意：すでに設定されている内容を絶対に変えてはいけない。大文字と小文字の違い、コロンとセミコロンの違いにも注意すること。この部分は特に慎重に行なう必要がある。

以上の設定変更を反映するため、①～③の作業で開いていたウィンドウを閉じて、新たに別のウィンドウを開いて⑤以下の作業を進める。

⑤C:¥MinGW¥msys¥1.0¥msys.bat (msys | Windows バッチファイル)をダブルクリックして実行する。
ターミナルが出てきたら以下のコマンドを実行する。

```
$ exit ↵          ☞exit とタイプして、エンターキーを押す
```

⑥ショートカットを作成する：
C:¥MinGW¥msys¥1.0¥msys.bat へのショートカットと
C:¥MinGW¥msys¥1.0¥home¥*username* へのショートカットを
デスクトップ上に作成する。
☞*username* は(半角英数字の)ユーザ名のことである。

⑦フォルダ files の中のファイル profile と emacs を
C:¥MinGW¥msys¥1.0¥home¥*username*¥の中にコピーする。

⑧ショートカットから msys を開始する。
☞「 msys.bat へのショートカット 」をダブルクリックする。
ターミナルが出てきたら以下のコマンドを実行する。

```
$ mv   profile   .profile ↵          ☞３つめの単語の語頭はピリオドである
$ mv   emacs   .emacs ↵
$ exit ↵
```

以上でインストールは完了である。再び「 msys.bat へのショートカット 」をダブルクリックすればターミナルが起動し、Emacs や gnuplot を利用できるはずである。

アンインストールについて
①～③で作成したフォルダを削除し、④で行なった環境変数の変更を元に戻せば、インストール前の状態に戻すことができる。