第２部　振り子の運動のシミュレーション

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？

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。

```
#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：

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

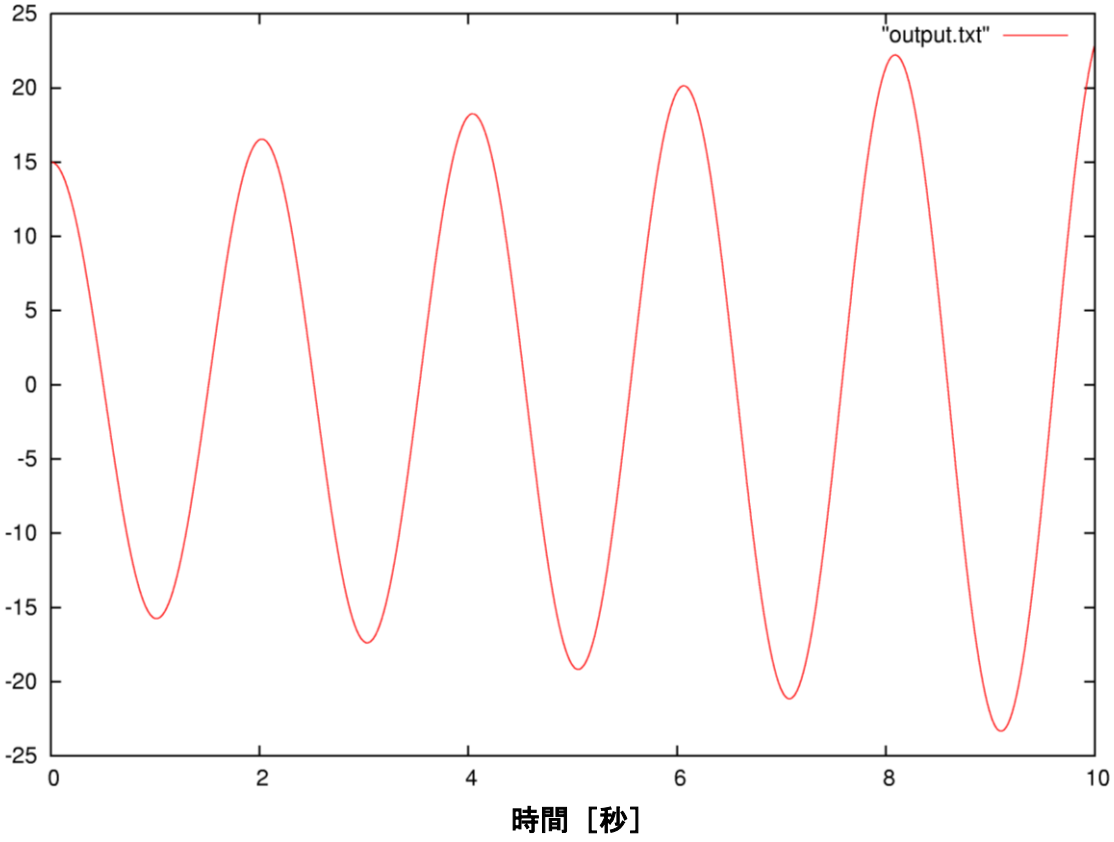
②とりあえず 10 秒後まで計算しよう

作業の手順：前回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,
           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;
}
```

振れ幅［度］



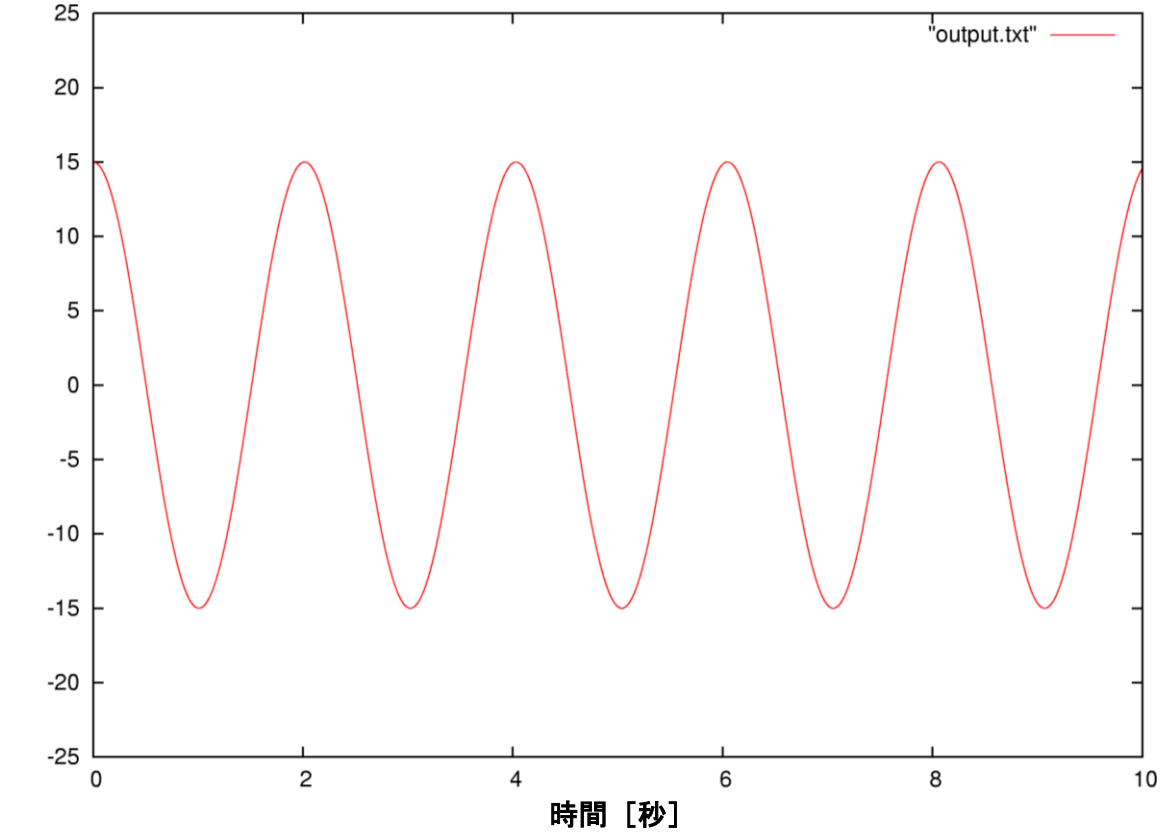
計算結果の例（長さ１ｍ、振幅 15° の場合）

③リープフロッグ法で計算しよう

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。
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）

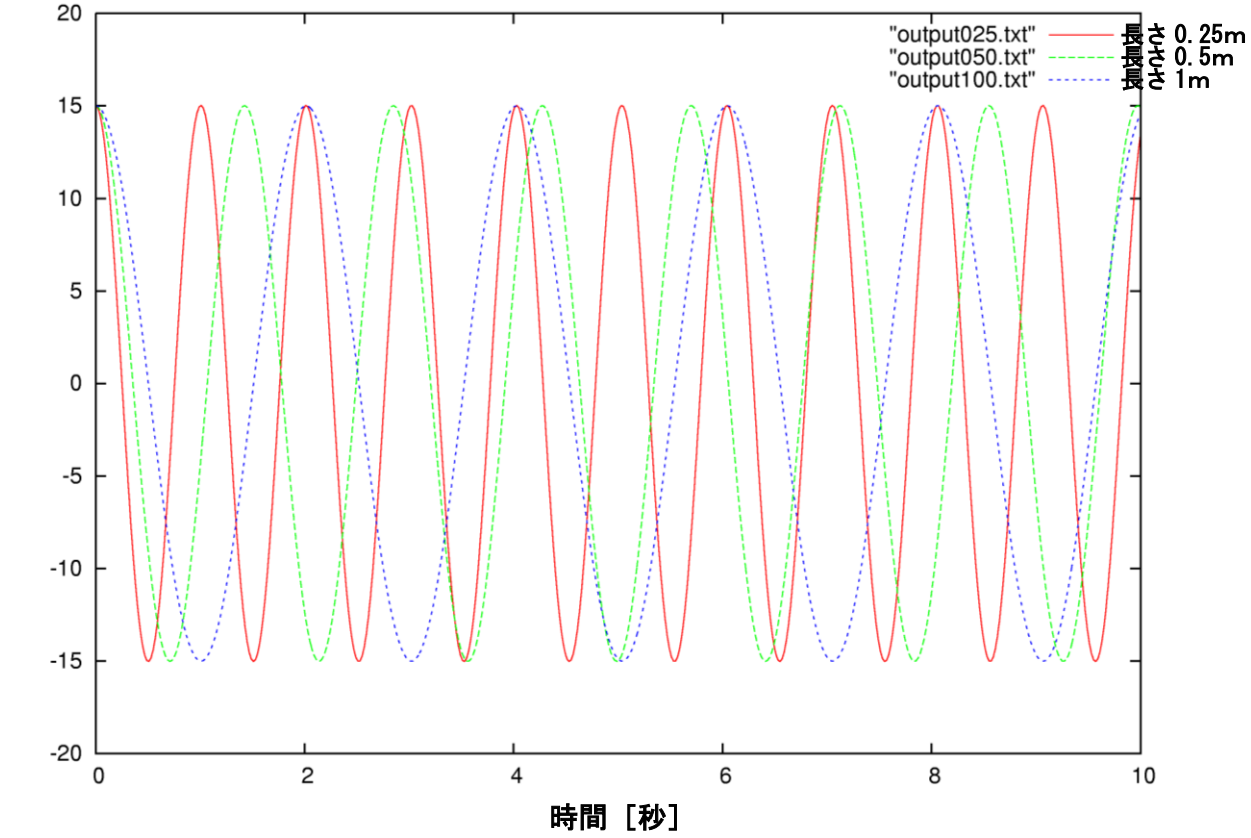
④実験してみよう

実験１：振幅を変えないで長さだけを変えてシミュレーションをしてみよう。

長さ１m、振幅 15°で実験　→　出力ファイル名を output100.txt に変更。
長さ 0.5m、振幅 15°で実験　→　出力ファイル名を output050.txt に変更。
長さ 0.25m、振幅 15°で実験　→　出力ファイル名を output025.txt に変更。
gnuplot で３つの出力ファイルをまとめて作図。

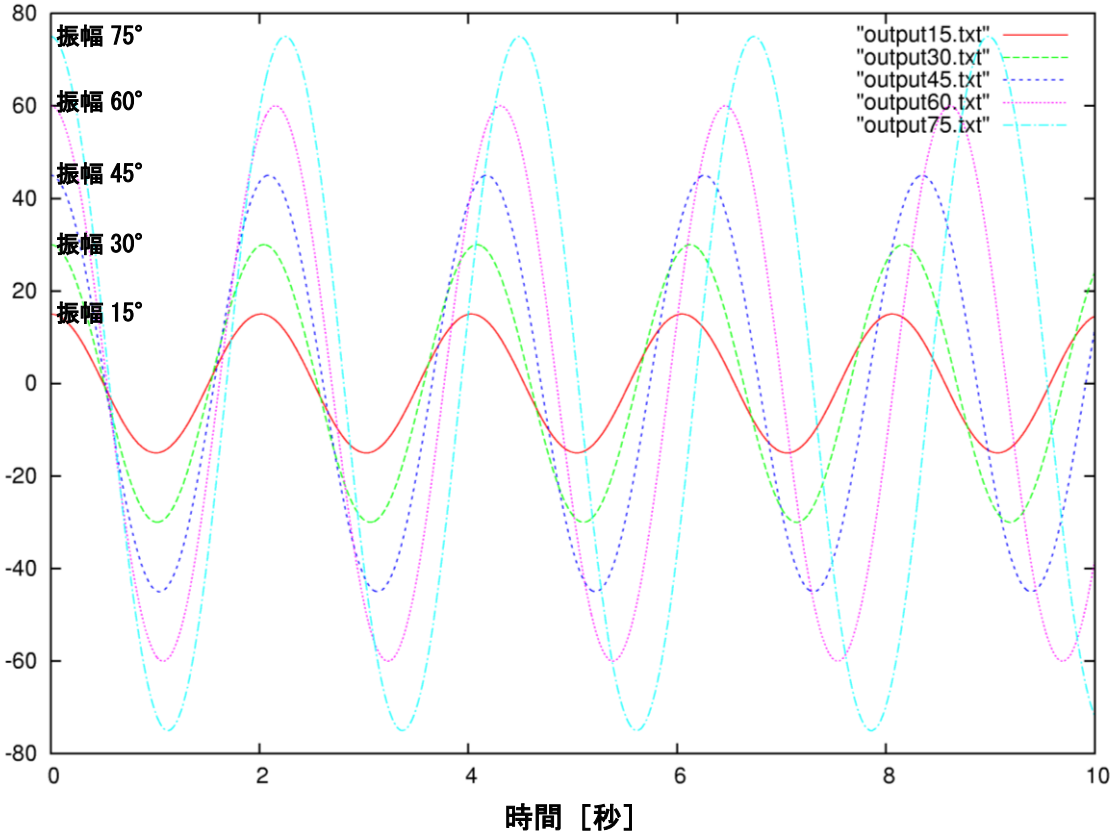
実験２：長さを変えないで振幅だけを変えてシミュレーションをしてみよう。

長さ１m、振幅 15°で実験　→　出力ファイル名を output15.txt に変更。
長さ１m、振幅 30°で実験　→　出力ファイル名を output30.txt に変更。
長さ１m、振幅 45°で実験　→　出力ファイル名を output45.txt に変更。
長さ１m、振幅 60°で実験　→　出力ファイル名を output60.txt に変更。
長さ１m、振幅 75°で実験　→　出力ファイル名を output75.txt に変更。
gnuplot で５つの出力ファイルをまとめて作図。



実験１の結果の例（振幅 15° の場合）

振れ幅［度］



実験２の結果の例（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。