物理学情報処理演習

10. 数値計算

数値計算

常微分方程式の解法

参考文献

Numerical Recipes in C

W. H. Press他著　　技術評論社

Advanced Engineering Mathematics

Erwin Keryszig　　John Wiley & Sons, Inc.

大久保晋

E-mail: buturi-johoshori@tiger.kobe-u.ac.jp

http://extreme.phys.sci.kobe-u.ac.jp/staffs/okubo/lectures/Programming/index.html

常微分方程式の数値解法

常微分方程式の解法は

１階の微分方程式の組に置き換えて考える

例えば２階の微分方程式

$$\frac{d^2y}{dx^2} + q(x)\frac{dy}{dx} = r(x)$$

は、新しい変数 z を補助的に用いて、連立の１階微分方程式

$$\frac{dy}{dx} = z(x)$$

$$\frac{dz}{dx} = r(x) - q(x)z(x)$$

に書き直すことができる。どのような常微分方程式についても当てはまる。

常微分方程式の数値解法

新しい変数の選び方は

お互いの変数のちょうど微分になるようにする

丸め誤差・オーバーフローの原因となる

特異な振る舞いを避ける

→　方程式に入っている他の因子や独立変数の累乗を新しい変数に組み込んでおくと良いこともある

解の中で本来の変数が滑らかであるのに対し、補助で用いた変数がおかしな振る舞いをするなら、

何故かを考え、異なる別の補助変数を選ぶ

常微分方程式の解法

- １階常微分方程式の解法を考える

$$\frac{dx}{dt} = f(x,t)$$

解は境界条件、$x(t_0)=x_0$ で決まる

- ２階常微分方程式の解法を考える

$$\frac{d^2x}{dt^2} + b\frac{dx}{dt} + cx = g(x,\dot{x},t)$$

この場合も解は境界条件で決まる。一般に境界条件は

–初期値問題

- 時刻 $t=t_0$ での $x=x_0$ 及び $dx/dt = v_0$ が与えられる

–境界値問題

- 時刻 $t=t_1$ 及び $t=t_2$ での $x(t_1)=x_1$、が$x(t_2)=x_2$ それぞれ与えられる

の２つに分類できる。

ここでは、簡単な初期値問題だけ考えることにする

常微分方程式の解法

２階の常微分方程式までを考えたが、一般の常微分方程式の問題は、関数 y_i, i=1, 2, …, N に対して

$$\frac{dy_i(x)}{dx} = f_i'(x, y_1, \cdots, y_N) \qquad i = 1, \cdots, N$$

の一般形をしたN元連立１階微分方程式に帰着する。

ただし、右辺の関数f'_iはわかっているものとする。（'は実際に微分するのでなく、関数f'がyの微分であるという意味）

初期値問題を解くルーチンの基礎にある考えは次の通り

上の方程式でdyとdxを有限なΔyとΔxに書き換え、その両辺にΔxをかける。こうして、独立変数xが刻み幅Δxだけステップしたときの関数の変化に体する代数式が与えられる。刻み幅を小さくした極限では元の微分方程式に対してよい近似となる。このやり方をEuler法と呼ぶ。

より上手な近似の取り方の方法としてRunge-Kutta法というのもある。

Euler法

下記の 1 階常微分方程式の解法を考える

$$\frac{dx}{dt} = f(x,t)$$

時刻t=t_0でx=x(t_0)=x_0 微小時刻Δt後のxの変化

$$\Delta x = x(t_0 + \Delta t) - x(t_0)$$

Δxを考えてみよう。x(t_0+Δt)をTaylor展開して

$$\Delta x = \Delta t \left.\frac{dx}{dt}\right|_{t=t_0} + \frac{(\Delta t)^2}{2} \left.\frac{d^2x}{dt^2}\right|_{t=t_0} + \cdots$$

Δtの 1 次まで考えて

$$\Delta x = f(x_0,t_0)\cdot\Delta t$$

これを利用して、時刻t=tnでの解xn=x(tn)を漸化式

$$t_{n+1} = t_n + \Delta t$$

$$x_{n+1} = x_n + \Delta t \cdot f(x_n,t_n)$$

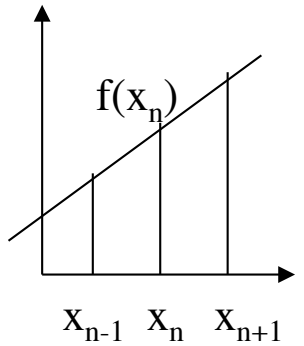
で求めることが出来る。この手順を陽的差分と呼ぶ。

これを（前進）Euler法と呼ぶ。

Euler法

先の方法は安定性の面で問題がある。下記の方程式(c:定数,c>0)を考える。

$$\frac{dx}{dt} = -cx$$



先のEulerの公式より

$$x_{n+1} = (1 - \Delta t \cdot c)x_n$$

となり、$\Delta t > 2/c$ の時に発散してしまう。

そこで、x_{n+1}での微分係数を使って（ x_{n+1} 位置の微係数をx_nとの差分 ）

$$t_{n+1} = t_n + \Delta t$$

$$x_{n+1} = x_n + \Delta t \cdot f(x_{n+1}, t_{n+1}) \qquad \cdots(1)$$

とする陰的差分を用いれば、

$$x_{n+1} = x_n \Big/ (1 + \Delta t \cdot c)$$

となり、安定化する。これを後退Euler法と呼ぶ。

しかし、後退Euler法は一般的には非線形方程式になって、

実際に解を見つけるのは困難

Euler法

そこで、通常次の様に線形化する。

$$x_{n+1} = x_n + \Delta t \cdot \left[f(x_n, x_{n+1}) + \left.\frac{\partial f}{\partial x}\right|_{x=x_n} \times \left(x_{n+1} - x_n\right) \right]$$

これを半陰的差分と呼び

$$x_{n+1} - x_n = \Delta t \cdot f(x_n, t_{n+1}) \left[1 - \Delta t \cdot \left.\frac{\partial f}{\partial t}\right|_{x=x_n} \right]^{-1}$$

から、x_{n+1}を求めていく

これと、前進及び後退Euler法からのΔtの２次まで考慮した

$$t_{n+1} = t_n + \Delta t$$

$$x_{n+1} = x_n + \frac{\Delta t}{2} \cdot \left[f(x_n, t_n) + f(x_{n+1}, t_{n+1}) \right]$$

を作ることができ、より精度の高い解を得ることができる。

式(1)に次式を用いる

$$\frac{f(x_{n+1}, t_{n+1}) - f(x_n, t_{n+1})}{x_{n+1} - x_n} = \left.\frac{\partial f}{\partial x}\right|_{x=x_n}$$

演習１０−１：Euler法

Euler法で１次の微分方程式（ただの微分）を解いてみよう

$$\frac{dx}{dt} = -15x$$

$$x(t=0) = 1$$

の解を求めるプログラムを作ってみる。

ここでは、t=0からt=10まで0.1間隔の情報を求めてみよう。ただしきざみ値は0.001とする。

正確な解は

$$x(t) = \exp(-15t)$$

であることは、明白である。

正確な解と、計算値の値をグラフ（OpenOfficeで）にして比較する。

次にきざみ値を0.1としてみよう。

演習１０−１：Euler法

$$\frac{dx}{dt} = -15x \qquad x(t=0) = 1$$

の漸化式は

$$t_{n+1} = t_n + \Delta t$$

$$x_{n+1} = x_n - 15x_n \cdot \Delta t = (1 - 15\Delta t) \cdot x_n$$

である。10-1.cのソースコードは以下の通り

```
#include <stdio.h>
#include <math.h>
                        きざみ値0.001
#define DELTA 1.0e-3
#define MAX_T 10.0
                          0< t <10.0

int main(void){
  double dt = DELTA;
  double t, x;
  int n;

  /* initial value */
  n = 0;
  t = 0.0;      初期値 t=0
  x = 1.0;        からスタート

printf("number¥tt¥tx¥texp(-15t)¥n");
  while (t<MAX_T){              t が MAX_T (=10.0)以下なら
                                          ループ続ける
    /* print out step status */
    if (n%100 == 0){          ループ100回に 1 度、結果出力
      /* printf("%8d: t=%12g x=%12g exact
        %12g ¥n", n, t, x, exp(-15.0*t)); */
      printf("%d¥t%g¥t%g¥t%g¥n",n, t, x,
              exp(-15.0*t));結果出力は
                                  回数, t, Euler計算結果, 厳密解
    }
    x *= (1.0 -15.0*dt);  x = x*(1.0 -15.0*dt)
    t += dt;      t = t + dt
    n += 1;       n = n + 1     nはループカウンター
  }
  return (0);
}
```

きざみ値0.1にするときは #define DELTA の値を変更する。さらに、今の出力は100回に一度の出力だが、1回に 1 度づつ出力させるように変更

演習１０−１：Euler法

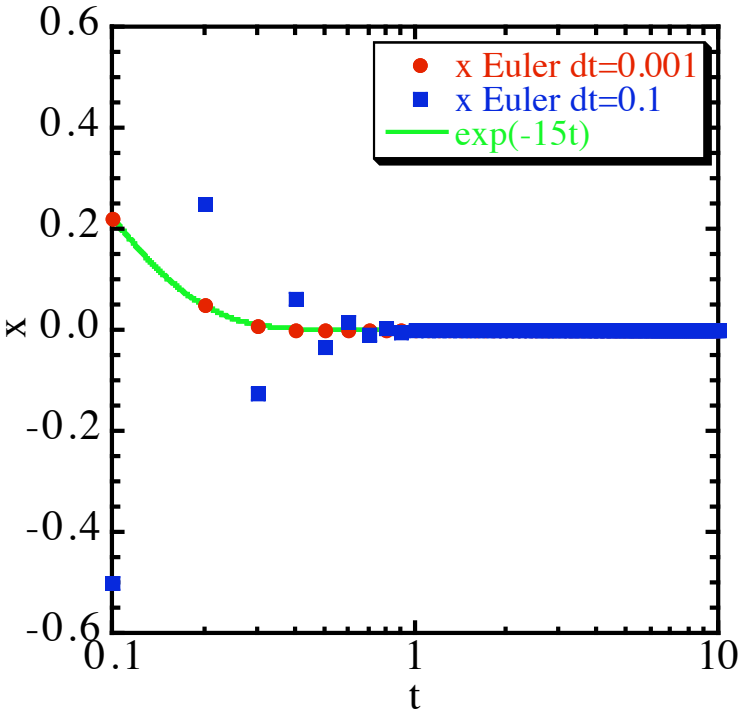
#define DELTA 1.0e-3　のとき、答えが以下のように出ただろうか？

```
$ gcc -lm -o 10-1 10-1.c ; ./10-1
number  t          x          exp(-15t)
0          0          1          1
100        0.1        0.220609          0.22313
200        0.2        0.0486683         0.0497871
300        0.3        0.0107367         0.011109
400        0.4        0.0023686         0.00247875
500        0.5        0.000522535       0.000553084
                 ・・・
9900       9.9        1.04396e-65       3.21565e-65
10000      10         2.30306e-66       7.1751e-66
```

左から計算ステップ数、変数tの値、
変数xの値（数値的に求めた微分方程式の解）、厳密解exp(-15t)の値である。計算ステップ数が小さいときは、求めたxは厳密解exp(-15t)にかなり近い値となっていることが分かる。

演習１０−１：Euler法

求めた計算結果をリダイレション（ > file名)でテキストファイルにしてOpenOfficeの表計算ソフトに入れてグラフにしよう。



きざみ値dt=0.001 のときと

きざみ値dt=0.1 のときで比較すると右図の様になっただろうか？

きざみ値0.1 のときは著しく理論曲線 exp(-15t) から逸脱してることが判る。

特に t < 1.0 では収束せず、上下にオーバーシュートしていることが判るであろう。

一方、きざみ値 0.001 では理論曲線（緑のライン）とよく一致していることがわかるであろう。

２階常微分方程式の解法

下記の２階常微分方程式の解法を考える。

$$\ddot{x} + b\dot{x} + cx = g(x, \dot{x}, t) \qquad \ddot{x} \equiv \frac{d^2x}{dt^2}, \dot{x} \equiv \frac{dx}{dt}$$

ここで、

$$y^{(1)} = x$$

$$y^{(2)} = \dot{x}$$

$$\vec{y} = \left(y^{(1)}, y^{(2)}\right)$$

とすると、上記の２階常微分方程式は

$$\dot{y}^{(1)} = f^{(1)}(\vec{y}, t) \equiv y^{(2)}$$

$$\dot{y}^{(2)} = f^{(2)}(\vec{y}, t) \equiv g(y^{(1)}, y^{(2)}, t) - by^{(2)} - cy^{(1)}$$

という連立１階常微分方程式に書き直すことができる。

この連立１階常微分方程式にEulerの式を適応して

$$t_{n+1} = t_n$$

$$\vec{y}_{n+1} = \vec{y}_n + \Delta t \cdot \vec{k}_0$$

$$k_0^{(i)} = f^{(i)}(\vec{y}_n, t_n)$$

を得る。

$$t_{n+1} = t_n$$

$$\begin{pmatrix} y^{(1)} \\ y^{(2)} \end{pmatrix}_{n+1} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \end{pmatrix}_n + \Delta t \cdot \begin{pmatrix} f^{(1)}_{(\vec{y}_n, t_n)} \\ f^{(2)}_{(\vec{y}_n, t_n)} \end{pmatrix}$$

Eulerの式

$\frac{dx}{dt} = f(x,t)$　のとき

$$t_{n+1} = t_n + \Delta t$$

$$x_{n+1} = x_n + \Delta t \cdot f(x_n, t_n)$$

で求められる。

２階常微分方程式の解法

具体的に書き下すと

$$
\begin{aligned}
y_{n+1}^{(1)} &= y_n^{(1)} + \Delta t \cdot f_{n(\vec{y}_n, t_n)}^{(1)} \\
&= y_n^{(1)} + \Delta t \cdot y_n^{(2)} \\
y_{n+1}^{(2)} &= y_n^{(2)} + \Delta t \cdot f_{n(\vec{y}_n, t_n)}^{(2)} \\
&= y_n^{(2)} + \Delta t \cdot \left(g(y_n^{(1)}, y_n^{(2)}, t) - b y_n^{(2)} - c y_n^{(1)} \right)
\end{aligned}
$$

n=0に対して、具体的に表せば

$$
\begin{aligned}
y_1^{(1)} &= y_0^{(1)} + \Delta t \cdot y_0^{(2)} \\
y_1^{(2)} &= y_0^{(2)} + \Delta t \cdot \left(g - b y_0^{(2)} - c y_0^{(1)} \right)
\end{aligned}
$$

で　$y^{(1)} = x$　、$y^{(2)} = \dot{x}$　を使えば

$$
\begin{aligned}
x_1 &= x_0 + \Delta t \cdot \dot{x}_0 \\
\dot{x}_1 &= \dot{x}_0 + \Delta t \cdot \left(g - b \dot{x}_0 - c x_0 \right)
\end{aligned}
$$

となる。初期条件として、　x_0　、　$\dot{x}_0$　が与えられていれば時間発展させれば、解を求めることが出来る。

演習１０−２：２階常微分方程式

質点をつなげたバネ振動の運動を解こう！

バネ定数k=10N/mのバネにつながれた、質量m=1kgの質点の運動を求めよう。質点はt=0で静止しており（v_0=0）、t=0でバネのつりあいの位置からx=2mの位置にあるとする。

求める常微分方程式（運動方程式）と初期条件は以下の通り

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

$$x(t=0) = 2.0[m]$$

$$v(t=0) = 0.0[m/s]$$

（前進）Euler法を用いてこの運動を解くことにする。

A)　t=0からt=5秒まで、0.0001秒のきざみ幅で出力は0.1秒ごとの解を求めてみる。

B)　B) t=0からt=5秒まで、0.1秒のきざみ幅で出力は0.1秒ごとの解を求めよ。　　もちろん、解析解は

$$x = 2\cos\left(\sqrt{\frac{k}{m}}\cdot t\right)$$

である。

演習１０−２：２階常微分方程式

x及びvを２つの変数とすると、前の方程式

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

$$x(t=0) = 2.0[m]$$

$$v(t=0) = 0.0[m/s]$$

は、次のように書ける。

$$\dot{x} = v$$

$$\dot{v} = -(10.0/1.0)\cdot x$$

Eulerの式を用いて以下の漸化式を得る

$$t_{n+1} = t_n + \Delta t$$

$$x_{n+1} = x_n + \Delta t \cdot v_n$$

$$v_{n+1} = v_n - 10.0 \cdot x_n \Delta t$$

Eulerの式

$$\frac{dx}{dt} = f(x,t)$$

のとき

$$t_{n+1} = t_n + \Delta t$$

$$x_{n+1} = x_n + \Delta t \cdot f(x_n, t_n)$$

で求められる。

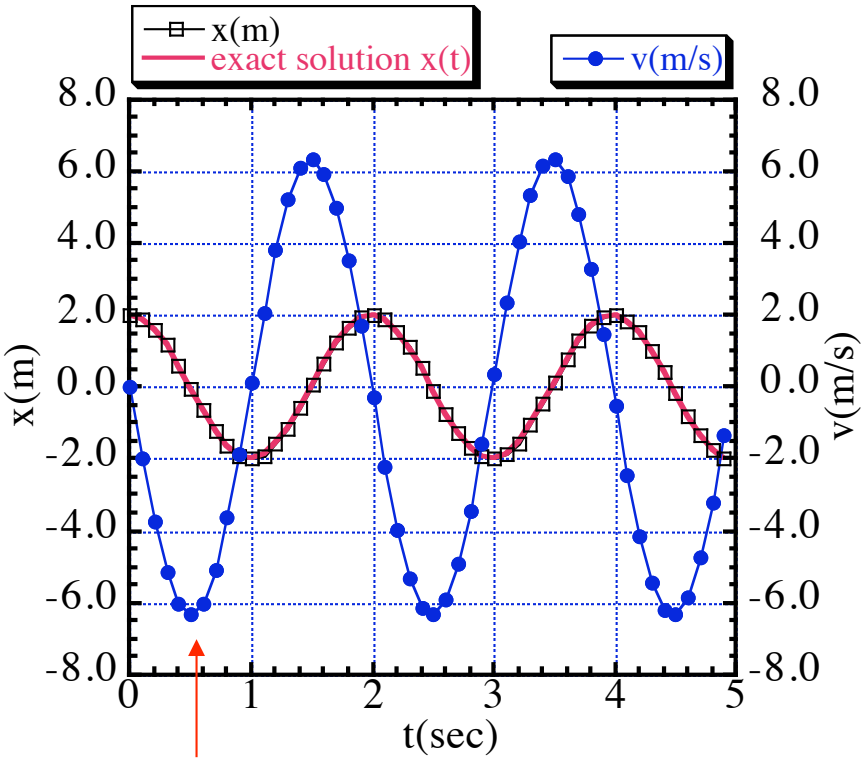
演習１０−２：２階常微分方程式

ソースプログラム10-2.cは以下のようになる。

```
#include <stdio.h>
#include <math.h>

                          きざみ値0.0001
#define DELTA 1.0e-4
#define MAX_T 5.0         0< t <5.0

const double k = 10.0;
const double m = 1.0;
const double a0 = 2.0;

double solution(double t);
                  解析解を与える関数の宣言
int main(void){
  double dt = DELTA;
  double t, x, v;
  double x0, v0;
  int n;

  /* initial value */
  n = 0;          初期値 t=0,
  t = 0.0;          x=a0=2.0 m,
  x = a0;           v=0.0 m/s
  v = 0.0;          からスタート

  printf("t(sec)¥tx(m)¥tv(m/s)¥t
                      exact solution x(t)¥n");
  while ( t<MAX_T ){ t が MAX_T (=5.0)以下ならループ続ける
   /* print out setp status */
    if ( n%1000 == 0 ){      ループ1000回に 1 度、結果出力
    /* printf("%8d: t=%g x=%g v=%g exact %g ¥n",
          n, t, x, v, solution(t)); */
      printf("%g¥t%g¥t%g¥t%g¥n", t, x, v,
             結果出力は              solution(t));
    }        t(sec), Euler計算結果 x(s), Euler計算結果 v(s) 解析解

   /* set current setp (x0, v0) info */
                  x0 に x を入れているので x0 は n回目のx
    x0 = x;       v0 も同様に vnです。
    v0 = v;

   /* calculate next step by using current step (x0, v0) */
                                  x = x + v0*dt
    x += (v0*dt);                          v = v -(k/m) * x0*dt
    v += ( (-k/m) * x0 * dt );  t = t + dt
    t += dt;                    n = n + 1   nはループカウンター
    n += 1;
  }
  return (0);
}

double solution(double t){
  return a0*cos(t*sqrt(k/m));
}
                  この関数は解析解を与える関数
```

演習１０−２：２階常微分方程式

計算結果をOpenOfficeでグラフ化してみよう。



バネの振動は単振動である。

釣り合いの位置（x=0）を中心に振動している解になっていることに注意する。

t=0, x=2.0mであるから、そこからスタートするのでx=cos(ωt)となっている。

一方 v はt=0で釣り合いの位置で最大速度なので、v=-sin(ωt)となっているだろうか？

きざみ値を0.0001としたので、数値解xと解析解 a0*cos(sqrt(k/m)t) は良い一致を示していることを確認しよう。

B) は各自やってみて、誤差がどのようになるか知ろう！

Euler法より良い方法：中点法

Euler法は言うなれば一次の近似である。Euler法を少し改良した方法が中点法と呼ばれる方法である。

Euler法では、始点（または終点）の微係数を使用したが、ステップを半分だけ進ませたt=Δt/2に対応する点の微係数を使用してたのが中点法である。

しかし、xの値は、始点の微係数を使用する。

$$t_{n+1} = t_n + \Delta t$$

$$x_{n+1} = x_n + \Delta t \cdot k_1$$

$$k_0 = f(x_n, t_n)$$

$$k_1 = f(x_n + k_0 \frac{\Delta t}{2}, t_n + \frac{\Delta t}{2})$$

とすればよい。この式は、厳密な解に対してΔtの２次まで正しい近似となっている。これを中点法（または２次のRunge-Kutta法）と呼ぶ。

Euler法より良い方法：中点法

中点法を用いて、物理によくありがちな２次方程式に応用してみる。

$$\ddot{x} = g(x)$$

ここで、 $v = \dot{x}$ とおいて、

$$t_{n+1} = t_n + \Delta t$$

$$x_{n+1} = x_n + \Delta t \cdot k_{x1}$$

$$v_{n+1} = v_n + \Delta t \cdot k_{v1}$$

とすればよい。ここで、始点、中点の微係数は

$$k_{x0} = v_n$$

$$k_{v0} = g(x_n)$$

$$k_{x1} = v_n + k_{v0}\frac{\Delta t}{2}$$

$$k_{v1} = g(x_n + k_{x0}\frac{\Delta t}{2})$$

となる。この関数 $\ddot{x} = g(x)$ は、２階微分が元のxの関数なので入れ子になっていることに注意。

各々の１階微分方程式にEulerの式を当てはめるが、その微係数は中点のものを使う。

中点： $x_n + f'(x_n)\frac{\Delta t}{2}$

各々の関数の中点の微係数を使う
２階微分はxの関数なので、その更に１階の微係数はv

演習１０−３：２階常微分方程式を中点法で

質点をつなげたバネ振動の運動を解こう！

バネ定数k=10N/mのバネにつながれた、質量m=1kgの質点の運動を求めよう。質点はt=0で
　　静止しており（v_0=0）、t=0でバネのつりあいの位置からx=2mの位置にあるとする。

求める常微分方程式（運動方程式）と初期条件は以下の通り

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

$$x(t=0) = 2.0[m]$$

$$v(t=0) = 0.0[m/s]$$

中点法を用いてこの運動を解くことにする。

A)　t=0からt=5秒まで、0.0001秒のきざみ幅で出力は0.1秒ごとの解を求めてみる。

B)　B) t=0からt=5秒まで、0.1秒のきざみ幅で出力は0.1秒ごとの解を求めよ。

もちろん、解析解は

$$x = 2\cos\left(\sqrt{\frac{k}{m}}\cdot t\right)$$

である。

演習１０－３：２階常微分方程式を中点法で

x及びvを２つの変数とすると、前の方程式

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

$$x(t=0) = 2.0[m]$$

$$v(t=0) = 0.0[m/s]$$

は、次のように書ける。

$$\dot{x} = v$$

$$\dot{v} = -(10.0/1.0)\cdot x$$

中点法を用いて以下の漸化式を得る

$$t_{n+1} = t_n + \Delta t$$

$$x_{n+1} = x_n + \left(v_n - 10x_n\frac{\Delta t}{2}\right)\Delta t$$

$$v_{n+1} = v_n - 10\left(x_n + v_n\frac{\Delta t}{2}\right)\Delta t$$

$$t_{n+1} = t_n + \Delta t$$

$$x_{n+1} = x_n + \Delta t\cdot k_{x1}$$

$$v_{n+1} = v_n + \Delta t\cdot k_{v1}$$

$$k_{x0} = v_n$$

$$k_{v0} = g(x_n)$$

$$k_{x1} = v_n + k_{v0}\frac{\Delta t}{2}$$

$$k_{v1} = g(x_n + k_{x0}\frac{\Delta t}{2})$$

結局、更に１階微分を入れる
vに対しては微分するとxになる
その更に微分はvだ。

演習１０－３：２階常微分方程式を中点法で

ソースプログラム10-3.cは以下のようになる。

```
#include <stdio.h>
#include <math.h>

                          きざみ値0.0001
#define DELTA 1.0e-4
#define MAX_T 5.0         0< t <5.0

const double k = 10.0;
const double m = 1.0;
const double a0 = 2.0;

double solution(double t);
                  解析解を与える関数の宣言

int main(void){
  double dt = DELTA;
  double t, x, v;
  double x0, v0;
  int n;

  /* initial value */
  n = 0;          初期値 t=0,
  t = 0.0;          x=a0=2.0 m,
  x = a0;           v=0.0 m/s
  v = 0.0;          からスタート

  printf("t(sec)¥tx(m)¥tv(m/s)¥t
                      exact solution x(t)¥n");
  while ( t<MAX_T ){ t が MAX_T (=5.0)以下ならループ続ける
   /* print out setp status */
    if ( n%1000 == 0 ){      ループ1000回に 1 度、結果出力
     /* printf("%8d: t=%g x=%g v=%g exact %g ¥n",
           n, t, x, v, solution(t)); */
      printf("%g¥t%g¥t%g¥t%g¥n", t, x, v,
             結果出力は              solution(t));
    }          t(sec), Euler計算結果 x(s), Euler計算結果 v(s) 解析解

   /* set current setp (x0, v0) info */
                x0 に x を入れているので x0 は n回目のx
                v0 も同様に vnです。
    x0 = x;
    v0 = v;                  x = x + (vn -(k/m)*xn (dt/2))*dt

   /* calculate next step by using current step (x0, v0) */
    x += ((v0 -(k/m)* x0 * (dt/2.0))*dt);
    v += ((-k/m)*(x0 + v0*(dt/2.0))*dt);
    t += dt;          t = t + dt    v = v -(k/m)* (xn + vn (dt/2))*dt
    n += 1;           n = n + 1   nはループカウンター
  }
  return (0);
}

double solution(double t){
  return a0*cos(t*sqrt(k/m));
}
                  この関数は解析解を与える関数
```

演習１０－３：２階常微分方程式を中点法で

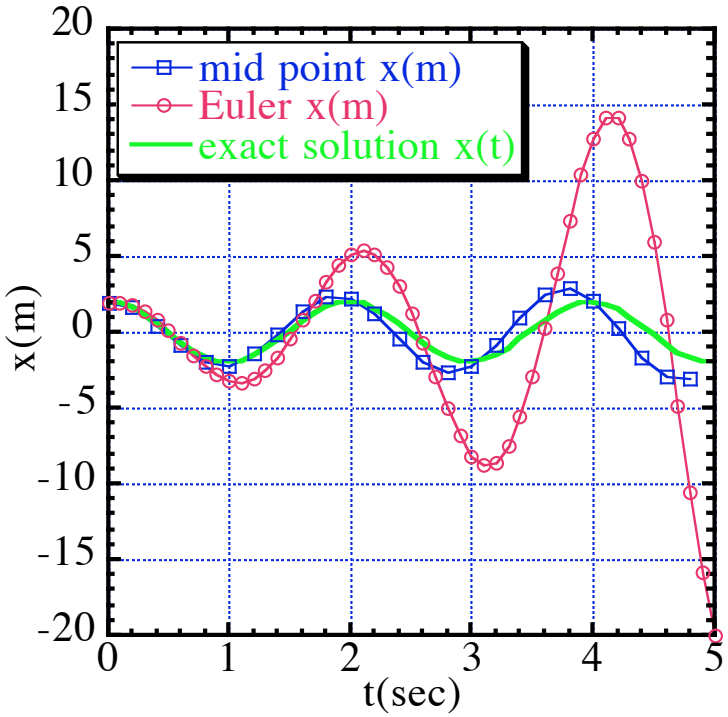
計算結果をOpenOfficeでグラフ化してみよう。

きざみ値 0.1のときの（前進）Euler法と中点法の比較を行ってみよ。

右図のようになっているだろうか？

バネの振動は単振動である。

緑のラインは解析解の結果である。釣り合いの位置（x=0）を中心に振動している解になっていることに注意する。



Euler法では、初期値から計算をするので、時間が経つにつれ誤差が増大している（赤線）。

中点法は3秒くらいまでは割と解析解に合っている（青線）。

中点法はある程度使えるということが判るが、Euler法は刻み値が大きくなければ使えない！

実際に常微分方程式の数値解を求めるには、きざみ値を変化させて答えが変化しないことを確認する必要がある！

Runge-Kutta法

常微分方程式の解を求めるには、一般にはRunge-Kutta法がよく用いられる。

これは、中点の始点と終点の微係数だけでなく、その中点の微係数も用いて、荷重平均をとったものである。

この方法は厳密な解に対してΔtの４次まで正しい近似式となる。

m個の連立常微分方程式

$$\frac{d\vec{y}}{dt} = \vec{f}(\vec{y},t) \quad \text{すなわち} \quad \dot{y}^{(i)} = f^{(i)}(\vec{y},t) \qquad i = 1 \cdots m$$

の解を求める漸化式は以下のように与えられる。

$$t_{n+1} = t_n + \Delta t$$

$$\vec{y}_{n+1} = \vec{y}_n + \frac{\Delta t}{6}\left(\vec{k}_0 + 2\vec{k}_1 + 2\vec{k}_2 + \vec{k}_3\right)$$

Runge-Kutta法

ここで

$$\vec{k}_0^{(i)} = f^{(i)}(\vec{y}_n, t_n)$$

$$\vec{k}_1^{(i)} = f^{(i)}(\vec{y}_n + \vec{k}_0 \cdot \frac{\Delta t}{2}, t_n + \frac{\Delta t}{2})$$

$$\vec{k}_2^{(i)} = f^{(i)}(\vec{y}_n + \vec{k}_1 \cdot \frac{\Delta t}{2}, t_n + \frac{\Delta t}{2})$$

$$\vec{k}_3^{(i)} = f^{(i)}(\vec{y}_n + \vec{k}_2 \cdot \Delta t, t_n + \Delta t)$$

である。