オイラー法

微分方程式を差分方程式にする：

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

のように微分方程式を、dt を有限にして評価する。つまり、最初t_0 秒の x の値が x_0 と分かっているとして、次の瞬間の x の値を求める。次の瞬間を、t_1秒目とすると、x の値を x_1 として・・・

$$x_1 - x_0 = f(x_0,t_0) \times (t_1 - t_0)$$

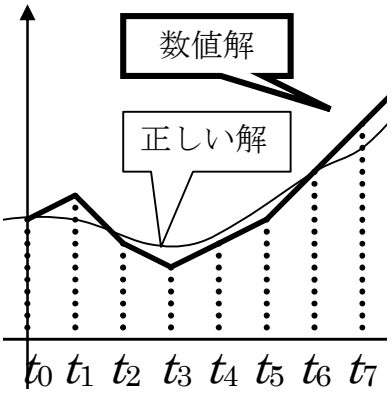
が成り立つとする。これをオイラー法という。従って、t_1秒目では、

$$x_1 = x_0 + f(x_0,t_0) \times (t_1 - t_0) \overset{t_1 - t_0 = \mathrm{STEP}}{=} x_0 + f(x_0,t_0) \times \mathrm{STEP}$$

と計算すればx_1が分かる。通常、時間の進みは一定として計算するので、便宜上、ここでは STEP としておく。実際は、STEP=0.01（秒）のような数値を取る。これが 0 に近いほど精密な結果が得られる。計算の結果は、

$$(x_0,t_0) \to (x_1,t_1) \to (x_2,t_2) \to \cdots \to (x_n,t_n)$$

の点の集合が得られるので、これをプロットすれば、xを求めることができる。つまり、



ところで、本当の微分式では、$x_1 \to x_0, t_1 \to t_0$になるので、$x_1 \to x_0, t_1 \to t_0$では

$$f(x_0,t_0) = f\left(\frac{x_0 + x_1}{2}, \frac{t_0 + t_1}{2}\right) = f\left(\frac{3x_0 + x_1}{4}, \frac{t_0 + 2t_1}{3}\right)$$

と同じになり、$x_1 - x_0 = f(x_0,t_0) \times (t_1 - t_0)$ でも $x_1 - x_0 = f\left(\frac{x_0 + x_1}{2}, \frac{t_0 + t_1}{2}\right) \times (t_1 - t_0)$ でも、$x_1 - x_0 = f\left(\frac{3x_0 + x_1}{4}, \frac{t_0 + 2t_1}{3}\right) \times (t_1 - t_0)$でも良いことになる。この微妙な点を使って改良したのをルンゲ・クッタ法という。

ルンゲ・クッタ法

微分方程式を差分方程式にする：

$$x_1 = x_0 + \frac{f_1 + 2f_2 + 2f_3 + f_4}{6} \times (t_1 - t_0) = x_0 + \frac{f_1 + 2f_2 + 2f_3 + f_4}{6} \times \mathrm{STEP}$$

$$f_1 = f(x_0, t_0)$$

$$f_2 = f\left(x_0 + \frac{f_1 \times \mathrm{STEP}}{2}, t_0 + \frac{\mathrm{STEP}}{2}\right)$$

$$f_3 = f\left(x_0 + \frac{f_2 \times \mathrm{STEP}}{2}, t_0 + \frac{\mathrm{STEP}}{2}\right)$$

$$f_4 = f(x_0 + f_3 \times \mathrm{STEP}, t_0 + \mathrm{STEP})$$

これは、もちろん、$x_1 \to x_0, t_1 \to t_0$ では STEP=0 なので、STEP=0 にすると

$$f_1 = f(x_0, t_0)$$

$$f_2 = f\left(x_0 + \frac{f_1 \times \mathrm{STEP}}{2}, t_0 + \frac{\mathrm{STEP}}{2}\right) \overset{\mathrm{STEP}=0}{\to} f(x_0, t_0)$$

$$f_3 = f\left(x_0 + \frac{f_2 \times \mathrm{STEP}}{2}, t_0 + \frac{\mathrm{STEP}}{2}\right) \overset{\mathrm{STEP}=0}{\to} f(x_0, t_0)$$

$$f_4 = f(x_0 + f_3 \times \mathrm{STEP}, t_0 + \mathrm{STEP}) \overset{\mathrm{STEP}=0}{\to} f(x_0, t_0)$$

なり、

$$\frac{f_1 + 2f_2 + 2f_3 + f_4}{6} = \frac{f(x_0, t_0) + 2f(x_0, t_0) + 2f(x_0, t_0) + f(x_0, t_0)}{6} = f(x_0, t_0)$$

になっている。

プログラム

これをプログラムするには、オイラー法では

$$x_1 = x_0 + f(x_0, t_0) \times \mathrm{STEP}$$

に対応して、まず、STEP を決めるが、例えば 0.05 にしてみよう。0.01 でもいくつでも良いが、小さい数値ほど精度が上がる。そこで、

```
#define STEP 0.05
```

と記述する。プログラム本体は、例えば、

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

の時には、

```
double f(double x, double t)
{
    // f(x,t) = sin xt
    return sin(x*t);
}
```

とすると・・・

```
// x_1 = x_0 + f(x_0, t_0) × STEP
double calclate_nextstep(double x, double t)
{
        double next_x;

        next_x = x + f(x, t)*STEP;

        return next_x;
}

int main(void)
{
    int i;
    double x[100], t; // 100 回の繰り返し

    x[0]=1.0; t=0.0; // 初期値(x[0]=1.0 だったりします)
    for(i=0; i<99; i++){ // 0-99 の 100 回の繰り返し
        printf("%f, %f¥n", t, x[i]); //ここを変えると表示
                                     //を変えれます
        x[i+1]= calclate_nextstep(x[i],t);
        t=t+STEP;
    }

    return 0;
}
```

でよい。

ルンゲ・クッタ法では$f(x_0,t_0)$をさらに分解するので、$f(x_0,t_0)$の代わりに$g(x_0,t_0)$として

$$x_1 = x_0 + g(x_0,t_0)\times(t_1-t_0) = x_0 + \frac{f_1+2f_2+2f_3+f_4}{6}\times \mathrm{STEP}$$

$$g(x_0,t_0) = \frac{f_1+2f_2+2f_3+f_4}{6}$$

にする。そして、

```
double f(double x, double t)
{
    //  f(x,t) = sin xt
    return sin(x*t);
}

double g(double x, double t)
{
    double f1, f2, f3, f4;

    // f1 = f(x0,t0)
    f1 = f(x, t);
    // f2 = f(x0 + f1×STEP/2, t0 + STEP/2)
    f2 = f(x+f1*STEP/2.0, t+STEP/2.0);
    // f3 = f(x0 + f2×STEP/2, t0 + STEP/2)X
    f3 = f(x+f2*STEP/2.0, t+STEP/2.0);
    // f4 = f(x0 + f3×STEP, t0 + STEP)
    f4 = f(x+f3*STEP, t+STEP);

    //  g(x0,t0) = (f1+2f2+2f3+f4)/6
    return (f1+2*f2+2*f3+f4)/6.0;
}
```

```
// x_1 = x_0 + g(x_0, t_0) × STEP
double calclate_nextstep(double x, double t)
{
 double next_x;

 next_x = x + g(x, t)*STEP;

 return next_x;
}

int main(void)
{
    int i;
    double x[100], t; // 100 回の繰り返し

    x[0]=1.0;
    t=0.0;
    for(i=0; i<99; i++){ // 0-99 の 100 回の繰り返し
        printf("%f, %f¥n", t, x[i]);  // ここを変えると表示
                                      //が変わります
        x[i+1]= calclate_nextstep(x[i],t);
        t=t+STEP;
    }

    return 0;
}
```