前期末試験解答用紙 (5E 計算機応用)

電気工学科　　学籍番号　　　　氏名

2006 年 9 月 28 日

1　ニュートン法 (Newton's Method)

[問 1] $\boxed{\textit{10 点}}$

閉区間 $[a, b]$ において，連続な関数 $f(x)$ の値が，

$$f(a)f(b) < 0 \tag{1}$$

の場合，$f(\alpha) = 0$ となる α がその区間にある．2 分法はこの性質を利用して近似解を求める．実際の数値計算のプログラムでは，区間 $[a, b]$ の中点 c を計算し，$f(a)f(c)$ と $f(b)f(c)$ のうち負になるほうを新たな区間 $[a, b]$ とする．c は新たな a または b になる．この操作を行う毎に，解が存在する区間の領域が半分になる．これを繰り返すことにより，任意の精度で方程式の近似解を求めることができる．これが，2 分法の計算原理である．

[問 2] $\boxed{\textit{20 点}}$

ニュートン法は、方程式 $f(x) = 0$ の近似解を求める方法の一つである。ある実数解を持つ関数 $f(x)$ をグラフにすると図のように書ける。この関数 $f(x)$ と x 軸の交点の x 座標がこの方程式の解となる。

ある近似解 x_n が求められたとすると、$(x_n, f(x_n))$ での接線が、x 軸と交わる点 x_{n+1} はさらに精度の良い近似解となる。そして、次の接線が x 軸と交わる点を次の近似解を x_{n+1} とする。図から分かるように、これを繰り返すと、非常に精度の良い近似解が得られる。

x_n と x_{n+1} の関係を示す漸化式は、接線の式

$$y - f(x_i) = f'(x_i)(x - x_i)$$

から求める。$y = 0$ の時の x の値が x_{i+1} なので、x_{i+1} は、

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

となる。これをニュートン法の漸化式である。

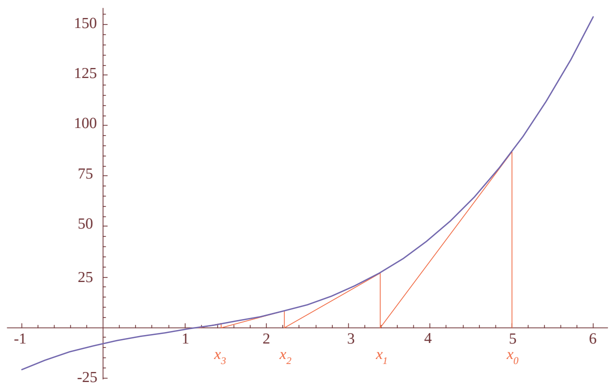

図 1: ニュートン法の収束

［**問 3**］ $\boxed{\textit{10 点}}$

方程式 $f(x)=0$ の真の解を α とする。x_{i+1} と真値 α の差の絶対値、誤差を計算する。これは、

$$
\begin{aligned}
|\alpha - x_{i+1}| &= \left|\alpha - x_i + \frac{f(x_i)}{f'(x_i)}\right| \\
&= \left|\alpha - x_i + \frac{f\left(\alpha + (x_i - \alpha)\right)}{f'\left(\alpha + (x_i - \alpha)\right)}\right| \\
&\qquad \alpha \text{ の周りでテイラー展開する。} \\
&= \left|\alpha - x_i + \frac{f(\alpha)}{f'(\alpha)} + \left[1 - \frac{f(\alpha)f''(\alpha)}{f'^2(\alpha)}\right](x_i - \alpha) + O\left((\alpha - x_i)^2\right)\right| \\
&\qquad f(\alpha) = 0 \text{ なので} \\
&= \left|O\left((\alpha - x_i)^2\right)\right|
\end{aligned}
$$

となる。二次の項が誤差の最大項である．これで，ニュートン法は二次収束であることが示された．

二次収束は，実際に計算回数に二乗で誤差が小さくなる．例えば，ニュートン法の 3 回の計算で誤差が 10^{-4} だったとすると，さらに 1 回漸化式を計算すると誤差は 10^{-8} になる．

［**問 4**］ $\boxed{\textit{10 点}}$

問 2 の答えである漸化式に $x_0 = 2$ を代入して x_1 を計算する．得られた x_1 を，さらに漸化式に代入すると，x_2 が得られる．これを計算するとき，$f(x) = x^2 - 2$，$f'(x) = 2x$ である．したがって，計算は以下のとおりである．

$$
\begin{aligned}
x_1 &= x_0 - \frac{x_0^2 - 2}{2x_0} = 2 - \frac{2^2 - 2}{2 \times 2} = 2 - \frac{1}{2} = \frac{3}{2} = 1.5000 \\
x_1 &= x_1 - \frac{x_1^2 - 2}{2x_1} = \frac{3}{2} - \frac{(\frac{3}{2})^2 - 2}{2 \times \frac{3}{2}} = \frac{3}{2} - \frac{1}{12} = \frac{17}{12} = 1.4166
\end{aligned}
$$

2　常微分方程式の数値計算法

［問 1］ $\boxed{7\text{点}}$

$$
\begin{aligned}
f(x_0+\Delta x) &= f(x_0)+f'(x_0)\Delta x+\frac{1}{2!}f''(x_0)\Delta x^2+\frac{1}{3!}f'''(x_0)\Delta x^3+\frac{1}{4!}f^{(4)}(x_0)\Delta x^4+\cdots \\
&= \sum_{n=0}^{\infty}\frac{1}{n!}f^{(n)}(x_0)\Delta x^n
\end{aligned}
$$

［問 2］ $\boxed{10\text{点}}$

微分方程式

$$
\frac{\mathrm{d}y}{\mathrm{d}x}=f(x,y)
$$

において，オイラー法はテイラー展開の二次以上の項を無視した数値計算法である．解を $y(x)$ とし，その漸化式を導くために $y_{i+1}=y(x_i+\Delta x)$ を考える．テーラー展開を使うと，

$$
\begin{aligned}
y_{i+1} &= y(x_i+\Delta x) \\
&= y(x_i)+\frac{\mathrm{d}x}{\mathrm{d}y}\Big|_{x=x_i}\Delta x+\frac{1}{2}\frac{\mathrm{d}^2y}{\mathrm{d}x^2}\Big|_{x=x_i}\Delta x^2+\ldots
\end{aligned}
$$

となる．このテイラー展開の二次以上の項を無視して，元の微分法定式を代入すると，

$$
y_{i+1}=y_i+f(x_i,y_i)\Delta x
$$

が得られる．これがオイラー法の漸化式である．

これを使って，微分方程式は

$$
x_i=i\Delta x \qquad y_{i+1}=y_i+f(x_i,y_i)\Delta x
$$

を繰り返し計算する．$Deltax$ は計算のステップ幅で，小さな適当な値を決める．通常 x_0 と y_0 は初期条件により与えられ，これから x_1 と y_1 を計算する．あとは順次計算を行えば，任意の x_i のときの y_i の値が分かる．

［問 3］ $\boxed{10\text{点}}$

計算のステップ幅を h とする。ホイン法は二次の精度なので、テイラー展開より

$$
\Delta y=y'(x_0)h+\frac{1}{2}y''(x_0)h^2
$$

となるようなアルゴリズムにする。

y の増分 Δy を計算するためには、1 階微分と 2 階微分の 2 項を満たす式が必要で、そのために、計算区間の両端の導関数の値を使う。この導関数は問題として与えられているので、計算は簡単である。そうして、区間の増分を α,β をパラメーターとした和で。

$$
\Delta y=h\{\alpha y'(x_0)+\beta y'(x_0+h)\}
$$

と表す。この式を x_0 の回りでテイラー展開し、3 次以降を無視すると

$$
\Delta y=(\alpha+\beta)y'(x_0)h+\beta y''(x_0)h^2 \tag{2}
$$

となる。これを二次までのテイラー展開の式と比較すると、$(\alpha,\beta)=(1/2,1/2)$ とすれば良いことが分かる。これから、

$$
\left\{
\begin{aligned}
k_1 &= hf(x_n,y_n) \\
k_2 &= hf(x_n+h,y_n+k_1) \\
y_{n+1} &= y_n+\frac{1}{2}(k_1+k_2)
\end{aligned}
\right. \tag{3}
$$

とすれば、2 次の精度が得られると類推できる。これがホイン法である。

3 プログラム作成

[問 1] $\boxed{20\text{点}}$

```
#include <stdio.h>
#include <math.h>
#define NSTEPS 2000                  // 計算ステップ数
#define NDIM 30000                   // 用意する配列の大きさ
#define X_MAX 2.0                    // 計算する最大 x

double f(double x, double y);        // プロトタイプ宣言

//=========================================================
// main 関数
//=========================================================
int main(void){
  double x[NDIM], y[NDIM];           // 計算結果を入れる
  double dx;
  int i;
  FILE *out;                         // 計算結果を保存するファイル


  dx = X_MAX/NSTEPS;                 // 計算のきざみ幅dxの計算
  x[0] = 0;
  y[0] = 0;


  //-------- オイラー法の計算 --------

  for(i=0; i<NSTEPS; i++){
    x[i+1] = x[0]+(i+1)*dx;          //x[i+1]=x[i]+dxよりも精度が良い
    y[i+1] = y[i]+f(x[i],y[i])*dx;
  }


  //-------- 計算結果をファイルに保存 --------

  out = fopen("result.txt","w");
  for(i=0; i<=NSTEPS; i++){
    fprintf(out,"%20.15f\t%20.15f\n", x[i],y[i]);
  }


  fclose(out);

  return 0;
}

//=========================================================
// dy/dx = f(x,y) の  f(x,y)を計算する関数
//=========================================================
double f(double x, double y){
  return x*x*sin(x)+sqrt(y)*cos(y);
}
```

[問 2] $\boxed{3\text{点}}$

ホイン法にするためには，次のように変数を用意する．

```
double k1, k2;
```

さらに，オイラー法の繰り返し計算の部分を以下のように書き直す．

```
for(i=0; i<NSTEPS; i++){
  k1 = dx*f(x[i], y[i]);
  k2 = dx*f(x[i]+dx, y[i]+k1);
  x[i+1] = x[0]+(i+1)*dx;
  y[i+1] = y[i]+0.5*(k1+k2);
}
```