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

2007 年 8 月 3 日

1　二分法

[**問 1**] 20 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**] ア: 20 点　　イ: 10 点

ア

```
while(b-a > EPS){
  c=(a+b)/2;
  if(func(c)*func(a) < 0){
      b=c;
    }else{
      a=c;
    }
}
```

イ

```
y = x*x-1-sin(x);
```

2　ニュートン法

[問 1] *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．

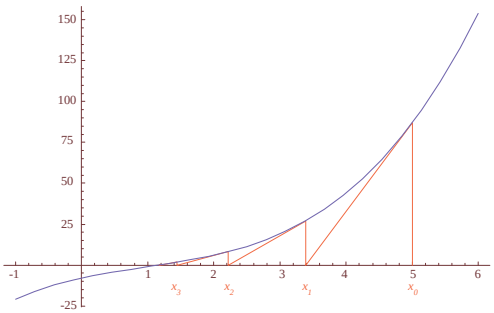

図 1: ニュートン法の収束

[問 2] *10 点*

方程式 $f(x)=0$ の真の解を α とする．真の解 α と $i+1$ 番目の近似解との x_{i+1} との差の絶対値，すなわち誤差は，

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

となる．これが，ニュートン法の近似解の収束を表す式である．

先に示した式の通り，ニュートン法の誤差は二乗で小さくなる．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 3] 20点

```c
#include <stdio.h>
#include <math.h>
#define IMAX 50
#define EPS (1.0e-15)              /* precision of calculation */

double func(double x);
double dfunc(double x);

/*=============================================================*/
/*        main function                                        */
/*=============================================================*/
int main(void)
{
  double x[IMAX+2];
  int i=-1;

  printf("\ninitial value x0 = ");
  scanf("%lf%*c", &x[0]);

  do{
    i++;
    x[i+1]=x[i]-func(x[i])/dfunc(x[i]);
  }while(i<=IMAX && fabs((x[i+1]-x[i])/x[i]) >= EPS);

  if(i>=IMAX){
    printf("\n  not converged !!! \n\n");
  }else{
    printf("\niteration = %d\nsolution  x = %20.15f\n\n",i,x[i+1]);
  }

  return 0;
}
/*=============================================================*/
/*        definition function                                  */
/*=============================================================*/
double func(double x)
{
  double y;

  y=x*x+x-cos(x);

  return y;
}
/*=============================================================*/
/*        definition derived function                          */
/*=============================================================*/
double dfunc(double x)
{
  double dydx;

  dydx=2*x+1+sin(x);

  return dydx;
}
```