４M　情報工学Ⅱ

第14回

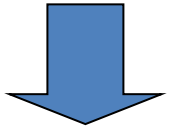
総合演習

授業予定

- ~~第１回　情報工学Ｉの復習~~
- ~~第２回　数値データの受け渡し~~
- ~~第３回　配列データの受け渡し~~
- ~~第４回　関数の設計・総合演習~~
- ~~第５回　ファイル書き込み~~
- ~~第６回　ファイル読み込み~~
- ~~第７回　総合演習~~
- ~~第８回　前期中間試験~~
- ~~第９回　試験の解答と解説~~
- ~~第１０回　ポインタの基礎~~
- ~~第１１回　ポインタと文字列~~
- ~~第１２回　構造体の宣言と利用~~
- ~~第１３回　構造体の配列的利用~~
- 第１４回　総合演習
- 第１５回　前期末試験

数値計算アルゴリズム

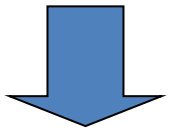
- 解析的に求められない式の計算
  - 非線形方程式の解
  - 積分で表示された式の値の計算
  - 円周率πの計算
- データ解析
  - 最小二乗法による実験データの近似式
  - フーリエ変換を用いたデータ解析
- 数値シミュレーション
  - 流体力学シミュレーション
  - 材料力学シミュレーション



数値計算の必要性

数値計算の特徴と限界

- 表現できる数値の有限性
  - 丸め誤差が生じる
  - 表現できるのは実数のみ
- 無限大・無限小が扱えない
  - 打切り誤差が存在する
- 計算能力・記憶領域の有限性
  - *効率的な計算方法が必要*



アルゴリズム

数値計算のアルゴリズム

1. 数値計算における誤差
2. 非線形方程式の解放
3. 連立一次方程式の解放
4. 多項式による補間
5. 数値積分法
6. 常微分方程式の解放
7. 行列の固有値問題

1．数値計算による誤差

1．1　誤差の種類

離散化誤差・・・無限回の処理を有限回へ

丸め誤差　・・・有限桁の数に置き換えで生じる

(i)固定小数点表示　:32bitの場合　-2^{31}～$2^{31}-1$

(ii)浮動小数点表示　:仮数部と指数部で表現

（例）$S=\Sigma_{i=1}^{n}(1/i^2)$ の計算結果
（単精度の場合）

$n=\infty$の真値　1.64493 0668

※n = 4097で情報落ち。nをそれ以上増やしても，和の値は増加しない。

n	計算結果（単精度）
1000	1.64393 4846
2000	1.64443 2068
3000	1.64459 4669
4000	1.64471 3879
4097	1.64472 5323
5000	1.64472 5323
10000	1.64472 5323

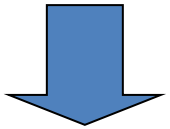
２．非線形方程式の解放

非線形方程式　$f(x)=0$ の根を求める

例えば,　　$f(x)=ax+b=0$　ならば　$x = -b/a$

しかし,　　$f(x)=(ax^2+b)\log x-c$　のような

非線形方程式では根は求まらない。



代表的アルゴリズム

２．１　二分法（bisection method）

２．２　反復法（iteration method）

２．３　ニュートン法（Newton method）

２．１　二分法

解を含む区間の中間点を求める操作を繰り返す

特徴：単純で必ず収束する。収束は遅い。

関数$f(x)$が連続で，２つの点a, b $(a < b)$で

$$f(a) < 0, f(b) > 0$$

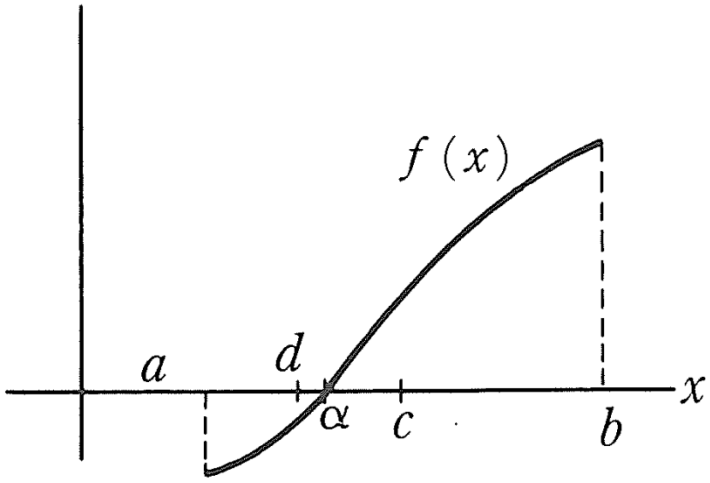
とする。根αは区間(a, b)の中に存在するので

a, bの中点

$$c = (a + b) / 2$$

の符号を調べる。

同様に中点の符号を繰り返し調べる。



2．2　反復法(不動点反復法)

$f(x)=0$　を

$$x = g(x)$$

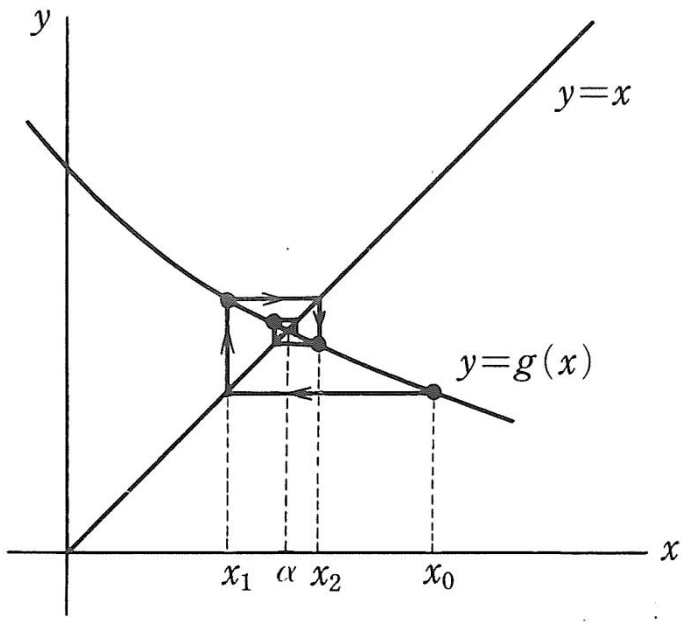
のように変形して，適当な初期値を定め

これを漸化式にした式

$$x_{k+1}=g(x_k),\ \ k=0,1,2,\cdots$$

によって近似値を求めていく。

特徴：収束が加速される。
ただし，必ずしも収束する
とは限らない。



2．3　ニュートン法

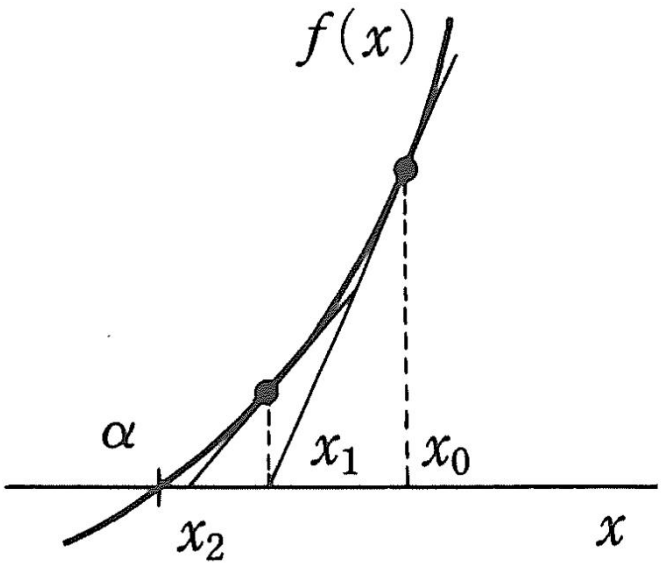
関数 $f(x)$ 上の点 x_0 において，曲線 $y = f(x_0)$ に**接線**を引き，接線と x 軸との交点 x_1 について，同様の操作を繰り返す．

特徴：収束は非常に速い。最も多く使われる。

表 3.4　$x^2 - a = 0$ をニュートン法で解いた場合（$a=3$，$x_0=3$）

k	x_k	e_k
0	3.000000E＋00	1.267950E＋00
1	2.000000E＋00	2.679490E－01
2	1.750000E＋00	1.794910E－02
3	1.732140E＋00	9.202960E－05
4	1.732050E＋00	0.000000E＋00

e_k：誤差



わずか3回で収束した

３．連立一次方程式の解放

３．１　ガウスの消去法

連立一次方程式から変数を消してゆく

３．２　LU分解法

行列式に変形してU（upper）とL（lower）に分解する

３．３　ガウス・ザイデル法（反復法）

初期値を適当に決めて繰り返す

３．１　ガウスの消去法

次のような連立一次方程式を考える。

$$2x+3y+3z = 5 \quad \cdots(1)$$

$$3x+2y-z = -4 \quad \cdots(2)$$

$$5x+4y+2z = 3 \quad \cdots(3)$$

手計算で解くときと同じ方法で解く。

１．１つの式を利用して，残り2式から x を消去する。

２．x が消去された2つの式から，y を消去する。

３．z のみを含んだ式からzを求め，残り2式から y と x を求める。

3．2　LU分解法

連立一次方程式を行列式で表す。

$$Ax=b$$

$$A=\begin{bmatrix} a_{11} & a_{12}\cdots\cdots a_{1n} \\ a_{21} & a_{22}\cdots\cdots a_{2n} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ a_{n1} & a_{n2}\cdots\cdots a_{nn} \end{bmatrix},\quad x=\begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_n \end{bmatrix},\quad b=\begin{bmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ b_n \end{bmatrix}$$

行列Aを上三角行列Uと下三角行列Lに分解して解く

$$A=L\cdot U$$

$$U=\begin{bmatrix} a_{11} & a_{12}\cdots\cdots\cdots a_{1n} \\ & a_{22}^{(1)}\cdots\cdots\cdots a_{2n}^{(1)} \\ & & a_{33}^{(2)}\cdots\cdots a_{3n}^{(2)} \\ 0 & & \cdots\cdots \\ & & \cdots\cdots \\ & & a_{nn}^{(n-1)} \end{bmatrix}\qquad L=\begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ l_{31} & l_{32} & 1 & 0 \\ \cdots\cdots & & & \\ \cdots\cdots & & & \\ l_{n1} & l_{n2}\cdots\cdots & l_{nn-1} & 1 \end{bmatrix}$$

３．３　ガウス･ザイデル法

次のような連立一次方程式を考える。

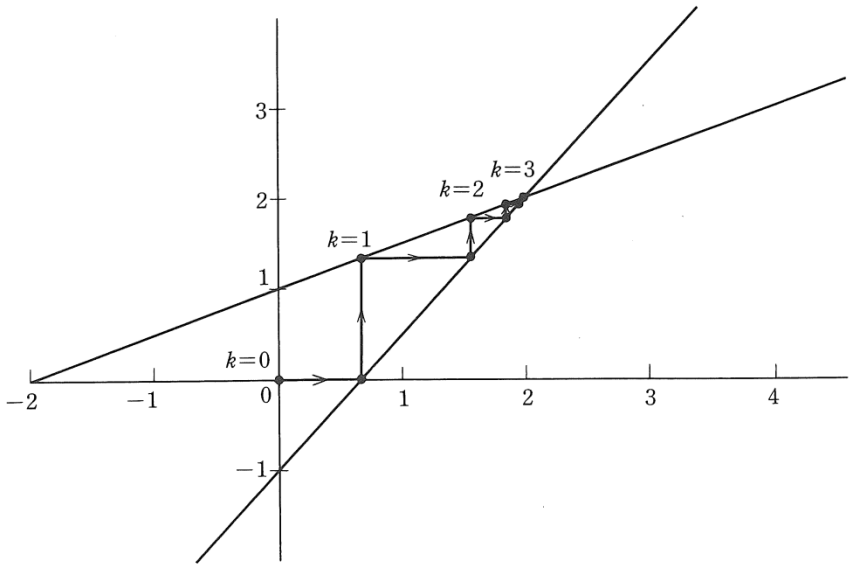
$$\begin{cases} 3x-2y=2 \\ x-2y=-2 \end{cases}$$

ここで，この方程式を変形して

$$x=(2+2y)/3$$

$$y=(2+x)/2$$

x と y の初期値を適当に決めて繰り返す。



４．多項式による補間

４．１　最小二乗法

データ点と近似式の残差の二乗が最小

４．２　ラグランジュ補間

ラグランジュ補間多項式によって近似

４．３　ニュートン補間

ニュートンの補間多項式によって近似

（ラグランジュ補間の補正）

4．1　最小二乗法

- データと多項式の残差の2乗和が最小

一次多項式（直線）を

$$P_1(x)=ax+b$$

とおくと，点x_iにおける残差は

$$e_i=f_i-P(x_i)$$

この２乗和は以下のとおり

$$E(a,b)=\sum_{i=1}^{n} e_i^2$$

$$=\sum_{i=1}^{n}(f_i-ax_i-b)^2$$

$$=\sum_{i=1}^{n}[f_i^2-2f_i(ax_i+b)+(ax_i+b)^2]$$

$$=(\sum_{i=1}^{n}x_i^2)a^2+2(\sum_{i=1}^{n}x_i)ab+(\sum_{i=1}^{n}1)b^2-2(\sum_{i=1}^{n}f_ix_i)a-2(\sum_{i=1}^{n}f_i)b+\sum_{i=1}^{n}f_i^2$$

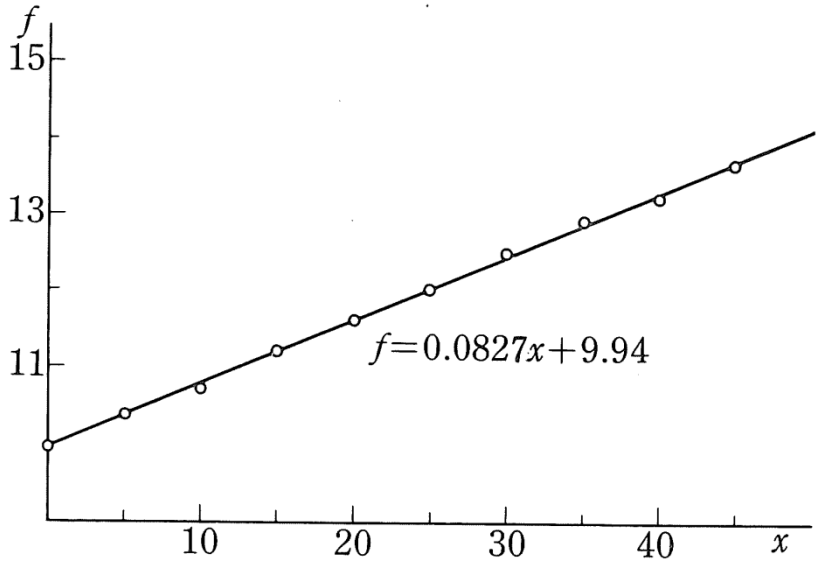


図 5.9　測定データと回帰直線の関係

４．２　ラグランジュ補間

- ラグランジュ補間関数

$$P_n(x) - f(x) = -\frac{f^{(n+1)}(\xi)}{(n+1)!}\,\phi(x)$$

$$\phi(x) = (x-x_0)\,(x-x_1)\cdots\cdots(x-x_n)$$

４．４　ニュートン補間

- ニュートン補間関数

$$P_k(x) = P_{k-1}(x) + f[x_k, x_{k-1}, \cdots, x_0](x-x_0)\,(x-x_1)\cdots(x-x_{k-1})$$

$$k = 1, 2, \cdots\cdots, n$$

５．数値積分法

５．１　台形公式

　　分割した台形の面積を足してゆく

５．２　シンプソン公式

　　台形公式を2次補間多項式で積分

５．３　その他

　　・ニュートン・コーツ公式

　　・ガウス・ルジャンドル公式

６．常微分方程式の解放

６．１　オイラー法

区間をN等分し，差分方程式（２点）で
逐次，近似解を求める。

６．２　ルンゲ・クッタ法

４点で重み付き平均を求めて精度を高く

６．３　その他

・アダムスの公式

多段階法（過去のステップを反映）

7. 行列の固有値

7．1　べき乗法

行列のべきをつくって反復する

7．2　ヤコビ法

行列を直交交換により対角化して，
直交交換を繰り返すことで対角行列に
収束させる方法

課題１９

二分法を用いて，非線形方程式の解を求めよ。
ただし，方程式は各自で適当に設定せよ。

（例） $f(x) = x^2 - a$ $(a=3,\ x_0=3)$

$f(x) = x^2 + x - a$ $(a=1,\ x_0=0.5)$

$f(x) = ax + \log x$ $(a=1,\ x_0=0.8)$

二分法のプログラム

```
1  #include <stdio.h>↓
2  #include <math.h>↓
3  ↓
4  /* 関数の定義 */↓
5  double f( double x )↓
6  {↓
7    return( pow(x,5.0) - 5.0*pow(x,3.0) + 4.0*x );↓
8  }↓
9  ↓
10 /* 2分法 */↓
11 double bisection( double a, double b, double eps )↓
12 {↓
13   double c;↓
14 ↓
15   do↓
16   {↓
17     c = (a+b)/2.0; ↓
18     if ( f(a)*f(c) < 0 )  ↓
19       b = c;↓
20     else ↓
21       a = c; ↓
22   }while( fabs(b-a) >= eps );↓
23   ↓
24   c = (a+b)/2.0; ↓
25 ↓
26   return c;↓
27 }↓
28 ↓
```

（つづき）

```
29 int main(void)↓
30 {↓
31   double a, b, x, h, y1, y2, eps = pow(2.0, -30.0);↓
32   int n;↓
33   ↓
34   printf("初期区間[a,b]を入力してください--->a b¥n");↓
35   scanf("%lf %lf", &a, &b );↓
36   printf("区間の分割数nを入力してください--->n¥n");↓
37   scanf("%d", &n );  ↓
38 ↓
39   h = (b-a)/n;  y1 = f(a);↓
40   for ( x = a+h ; x <= b ; x += h)↓
41   {↓
42     y2 = f(x); ↓
43     if ( y1*y2 < 0.0 )  ↓
44     {↓
45       printf("求める答えはx=%fです¥n", bisection(x-h, x, eps) );↓
46     }↓
47     y1 = y2;↓
48   }↓
49   return 0;↓
50 }↓
51 [EOF]
```

課題20

ニュートン法を用いて，二分法で用いたのと同じ非線形方程式の解を求めて収束回数を比較せよ。

（例） $f(x) = x^2 - a$ $(a=3,\ x_0=3)$

$f(x) = x^2 + x - a$ $(a=1,\ x_0=0.5)$

$f(x) = ax + \log x$ $(a=1,\ x_0=0.8)$

ニュートン法のプログラム

```
1  #include <stdio.h>↓
2  #include <math.h>↓
3  ↓
4  #define EPS  pow(10.0,-8.0)      /* epsilonの設定 */↓
5  #define NMAX 100                 /* 最大反復回数 */↓
6  ↓
7  /* Newton法(重複次数つき) */↓
8  void newton1( double x )↓
9  {↓
10   int n=0; double d, x0=x, x1=x, x2=x, m; /* xでx0,x1,x2を初期化 */↓
11 ↓
12   do↓
13   {↓
14     d = -f(x)/df(x);↓
15     x = x + d;↓
16     n ++;↓
17     x0 = x1; x1 = x2; x2 = x; ↓
18   }while( fabs(d) > EPS && n < NMAX);↓
19 ↓
20   m = (x0 - x1)/(x0 - 2.0*x1 + x2) ; /* 重複次数の計算 */↓
21   if ( n == NMAX )↓
22   {↓
23     printf("答えが見つかりませんでした¥n");↓
24   }↓
25   else↓
26   {↓
27     printf("答えはx=%fで,重複次数は%fです¥n",x, m);↓
28   }↓
29 }↓
30 ↓
```

（つづき）

```
31 /* 関数の定義 */↓
32 double f(double x)↓
33 {↓
34   return( x*x*x - x*x - x + 1.0 );↓
35 }↓
36 ↓
37 /* 導関数の定義 */↓
38 double df(double x)↓
39 {↓
40   return( 3.0*x*x - 2.0*x - 1.0 );↓
41 }↓
42 ↓
43 int main(void)↓
44 {↓
45   double x;↓
46   printf("初期値x0を入力してください¥n");↓
47   scanf("%lf",&x);↓
48 ↓
49   newton1( x );↓
50 ↓
51   return 0;↓
52 }↓
53 [EOF]
```