

## スライド1 講義アウトライン

- 関数「 $f(x)$ 」と「 $g(x)$ 」
  - 「 $f(x)$ 」近似による関数近似
  - 「 $g(x)$ 」補間

## スライド2 関数近似 p.116

- 複雑な関数を簡単な関数で近似する 関数「 $f(x)$ 」
- 閉区間 $[a, b]$ で定義された関数 $f(x)$ を  $g(x) = \sum a_i \phi_i(x)$  で近似する. 関数系  $\phi_i(x)$  は $[a, b]$ 上で連続かつ1次独立関数が1次独立とは,

$$c_0 \phi_0(x) + c_1 \phi_1(x) + \cdots + c_{n-1} \phi_{n-1}(x) = 0 \rightarrow c_0 = c_1 = \cdots = c_{n-1} = 0$$

関数の差の「 $L_2$ ノルム」  $\|f - g\|_2^2 := \int_a^b \{f(x) - g(x)\}^2 dx$

$$= \int_a^b \left\{ f(x) - \sum_{i=0}^{n-1} a_i \phi_i(x) \right\}^2 dx$$

が最小になるように係数  $a_i$  を定める.

## スライド3 定理6.1 p.117

- (証明) (6.1)より

$$\begin{aligned} F(a_0, a_1, \dots, a_{n-1}) &:= \|f - g\|_2^2 = (f - g, f - g) = (f, f) - 2(f, g) + (g, g) \\ &= \|f\|_2^2 - 2\left(f, \sum_{i=0}^{n-1} a_i \phi_i\right) + \left(\sum_{i=0}^{n-1} a_i \phi_i, \sum_{j=0}^{n-1} a_j \phi_j\right) \\ &= \|f\|_2^2 - 2\left(f, \sum_{i=0}^{n-1} a_i \phi_i\right) + \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i a_j (\phi_i, \phi_j) \end{aligned}$$

が最小になるには「 $L_2$ ノルム」の「 $L_2$ ノルム」条件を満たさなければならない.

$$\frac{\partial F}{\partial a_i} = -2(f, \phi_i) + 2 \sum_{j=0}^{n-1} a_j (\phi_i, \phi_j) = 0$$

連立1次方程式「 $L_2$ ノルム」を解くことと同値.

スライド4 グラム行列式

- 要素  $x_0, x_1, \dots, x_{n-1}$  が線形独立であることと、以下の「 $\Delta_{n-1}$ 」行列式が0でないことと同値。  
(6.2) はただ「 $\Delta_{n-1} \neq 0$ 」の解を持つ。

$$\Delta_{n-1} = \begin{vmatrix} (x_0, x_0) & (x_0, \phi_1) & \cdots & (x_0, x_{n-1}) \\ (x_1, x_0) & (x_1, \phi_1) & \cdots & (x_1, x_{n-1}) \\ \vdots & \vdots & \cdots & \vdots \\ (x_{n-1}, x_0) & (x_{n-1}, \phi_1) & \cdots & (x_{n-1}, x_{n-1}) \end{vmatrix} \neq 0$$

定理 6.2  $\phi_i = x_i (i=0,1,2,\dots)$  は  $[a,b]$  で連続であり、かつ1次独立である。

(証明) 任意の自然数  $n$  に対して、

$$a_0 + a_1x + a_2x^2 + \cdots + a_nx^n = 0$$

と仮定すると、閉区間  $[a, b]$  の任意の  $t$  に対して0、したがって

$$a_0 = a_1 = a_2 = \cdots = a_n = 0$$

なぜならば、 $n$  次方程式の解は高々  $n$  個。

スライド5 関数近似の例 p.118

- 例 6.3  $f(x)=x^2$  に対する最小2乗近似1次式  $g(x)=a_0 + a_1x$  を  $[0,1]$  で求めよ。

$$(\phi_0, \phi_0) = \int_0^1 1dx = 1 \quad (\phi_0, \phi_1) = \int_0^1 xdx = \frac{1}{2} = (\phi_0, \phi_1)$$

$$(\phi_1, \phi_1) = \int_0^1 x^2dx = \frac{1}{3}$$

$$(f, \phi_0) = \int_0^1 x^2dx = \frac{1}{3} \quad (f, \phi_1) = \int_0^1 x^3dx = \frac{1}{4}$$

$$\begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{4} \end{bmatrix}$$

- $\phi_0(x)=1, \phi_1(x)=x$  とすると

したがって、 $a_0 = -1/6, a_1 = 1$ 。よって  $y=x-1/6$ 。

スライド6 定理 6.4 p.119

$m$  組の与えられたデータ  $(x_0, y_0), (x_1, y_1), \dots, (x_{m-1}, y_{m-1})$  を通る関数  $f(x)$  を  $g(x) = \sum a_i \phi_i(x)$  によって近似することを考える。

$$F(a_0, a_1, \dots, a_{n-1}) := \|f - g\|_2^2 = \sum_{k=0}^{m-1} (y_k - \sum_{j=0}^{n-1} a_j \phi_j(x_k))^2$$

が最小になるように  $a_0, a_1, \dots, a_{n-1}$  を決定する。

連立1次方程式

の解である。

$$\begin{bmatrix} (\phi_0, \phi_0) & (\phi_0, \phi_1) & \cdots & (\phi_0, \phi_{n-1}) \\ (\phi_1, \phi_0) & (\phi_1, \phi_1) & \cdots & (\phi_1, \phi_{n-1}) \\ \vdots & \vdots & \cdots & \vdots \\ (\phi_{n-1}, \phi_0) & (\phi_{n-1}, \phi_1) & \cdots & (\phi_{n-1}, \phi_{n-1}) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} (y, \phi_0) \\ (y, \phi_1) \\ \vdots \\ (y, \phi_{n-1}) \end{bmatrix}$$

$$(y, \phi_i) = \sum_{k=0}^{m-1} y_k \phi_i(x_k)$$

$$(\phi_i, \phi_j) = \sum_{k=0}^{m-1} \phi_i(x_k) \phi_j(x_k) \quad (k = 0, 1, \dots, m-1; i, j = 0, 1, \dots, n-1)$$

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define M 6 /* データのペア数 */
#define N 3 /* N次式で近似 */
/* ベクトルの入力 */
void input_vector2( double *b, char c, int n, FILE *fin, FILE *fout);
/* 部分ピボット選択付きガウス消去法 */
void gauss2( double a[N+1][N+1], double b[N+1], int n );
/* 最小2乗近似 */
void least_square( double *x, double *y, FILE *fout );
int main(void)
{
    FILE *fin, *fout;
    double x[M], y[M];
    /* ファイルのオープン */
    if ( (fin = fopen("input_func.dat", "r")) == NULL )
    {
        printf("ファイルが見つかりません : input_func.dat %n");
        exit(1);
    }
    if( (fout = fopen( "output_func.dat", "w")) == NULL )
    {
        printf("ファイルが作成できません : output_func.dat %n");
        exit(1);
    }
    input_vector2( x, 'x', M, fin, fout ); /* ベクトル x の入出力 */
    input_vector2( y, 'y', M, fin, fout ); /* ベクトル y の入出力 */
    least_square( x, y, fout ); /* 最小2乗近似 */
    fclose(fin); fclose(fout); /* ファイルのクローズ */
    return 0;
}

void least_square( double x[M], double y[M], FILE *fout )
{
    double a[N+1], p[N+1][N+1];
    int i, j, k;
    /* 右辺ベクトルの作成 */
    for(i=0; i <= N; i++) {
        a[i]=0.0;
        for( j = 0; j < M; j++) {
            a[i] += y[j]*pow(x[j],(double)i);
        }
    }
    /* 係数行列の作成 */
    for( i = 0; i <= N; i++) {
        for( j = 0; j <= i; j++) {
            p[i][j]=0.0;
            for( k = 0; k < M; k++) {
                p[i][j] += pow( x[k], (double)(i+j) );
            }
            p[j][i] = p[i][j];
        }
    }
    /* 連立1次方程式を解く。結果は a に上書き */
    gauss2( p, a, N+1 );
    /* 結果の出力 */

```

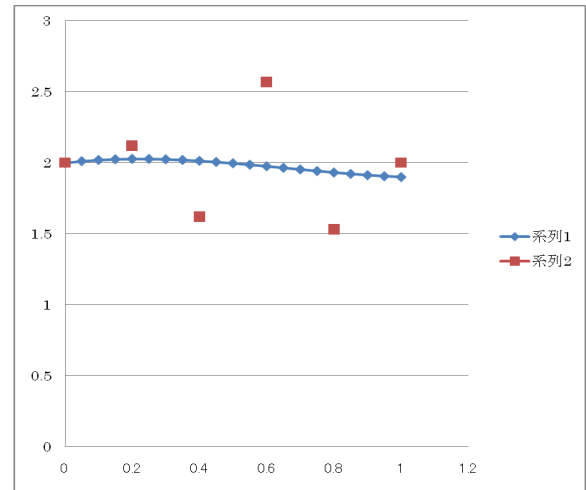
input_func.dat					
0.0	0.2	0.4	0.6	0.8	1.0
2.0	2.12	1.62	2.57	1.53	2.0

output_func.dat					
ベクトル x は次の通りです					
0.00	0.20	0.40	0.60	0.80	1.00
ベクトル y は次の通りです					
2.00	2.12	1.62	2.57	1.53	2.00
最小2乗近似式は					
y = 0.38 x^3 - 0.76 x^2 + 0.28 x^1 + 2.00 x^0					

```

fprintf(fout, "最小 2 乗近似式は y =");
for(i = N; i >= 0; i--) {
    if(a[i]>0){
        if(i==N){
            fprintf(fout, " %5.2f x^%d ", a[i],i);
        }
        else{
            fprintf(fout, "+ %5.2f x^%d ", a[i],i);
        }
    }
    else{
        fprintf(fout, "- %5.2f x^%d ", fabs(a[i]),i);
    }
}
fprintf(fout, "\n");
}

```



スライド 8 ラグランジュ補間 p.124

点  $(x_0, f_0), (x_1, f_1), \dots, (x_n, f_n)$  が与えられたとき、これらすべての点  $(x_i, f_i)$  を通る曲線  $y=f(x)$  を求めて  $x_0 < x < x_n$  の与えられた点 以外の関数値を求めることを「 $\quad$ 」, 「 $\quad$ 」 (interpolation) するという。  $f_k = f(x_k), k=0,1,\dots,n$  が与えられたとき、等式  $P(x_k) = f_k, k=0,1,\dots,n$  を満たす多項式  $P(x)$  を  $f(x)$  の「 $\quad$ 」式という。

定理 6.5 補間条件を満たす  $n$  次多項式  $P_n(x)$  はただ 1 つに定まる。

$$P_n(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$$

$$V = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{bmatrix}$$

$V$  の行列式: 「 $\quad$ 」 (Vandermonde) の行列, 解が 「 $\quad$ 」 に定まる。

スライド 9 ラグランジュ補間 p.125

$n$  次のラグランジュ補間多項式, ラグランジュ補間(Lagrange interpolation)

$$l_i(x) = \prod_{k=0, k \neq i}^n \frac{x - x_k}{x_i - x_k}$$

$$P_n = \sum_{i=0}^n f_i l_i(x)$$

基本多項式  $l_i(x)$  の点  $x_k$  ( $0 \leq k \leq n$ ) での値は

$$l_i(x_k) = \delta_{ik} := \begin{cases} 1 & (k = i) \\ 0 & (k \neq i) \end{cases}$$

$$P_n(x_k) = \sum_{i=0}^n f_i l_i(x_k) = \sum_{i=0}^n f_i \delta_{ik} = f_k$$

```

#include <stdio.h>
#include <stdlib.h>
#define N 9
/* ベクトルの入力 */
void input_vector3( double b[N+1], char c, FILE *fin );
/* ラグランジュ補間 */
double lagrange( double x[N+1], double y[N+1], double xi );

int main(void)
{
    FILE *fin, *fout;
    double x[N+1], y[N+1], xi; /* xiは補間点 */
    printf("補間点を入力してください--->");
    scanf("%lf", &xi);
    /* ファイルのオープン */
    if ( (fin = fopen( "input_lag.dat", "r" )) == NULL ){
        printf("ファイルが見つかりません : input_lag.dat ¥n");
        exit(1);
    }
    if( (fout = fopen( "output_lag.dat", "w" )) == NULL )
    {
        printf("ファイルが作成できません : output_lag.dat ¥n");
        exit(1);
    }
    input_vector3( x, 'x', fin ); /* ベクトルxの入出力 */
    input_vector3( y, 'y', fin ); /* ベクトルyの入出力 */
    printf("補間の結果は、P(%f)=%f¥n", xi, lagrange(x,y,xi) );

    /* グラフを描くために結果をファイルに出力 */
    for( xi = x[0]; xi <= x[N]; xi += 0.01 ){
        fprintf(fout, "%f ¥t %f¥n", xi, lagrange(x,y,xi) );
    }
    fclose(fin); fclose(fout); /* ファイルのクローズ */
    return 0;
}

/* ラグランジュ補間 */
double lagrange( double x[N+1], double y[N+1], double xi )
{
    int i, k;
    double pn = 0.0, li;

    /* P_n(x)の計算 */
    for ( i = 0; i <= N; i++){
        li = 1.0;
        /* l_i(x)の計算 */
        for(k = 0; k <= N; k++){
            if( k != i ) li *= (xi -x[k]) / (x[i]-x[k]);
        }
        pn += li * y[i];
    }

    return pn;
}

/* b[0...N]の入力 */
void input_vector3( double b[N+1], char c, FILE *fin )
{
    int i;
    for(i = 0; i <= N; i++)
    {
        fscanf(fin, "%lf", &b[i]);
    }
}

```

input_lag.dat									
0.0	0.2	0.4	0.6	0.8	1.2	1.4	1.6	1.8	2.0
2.0	2.1	1.6	2.6	1.5	2.7	0.67	3.5	0.94	2.0

