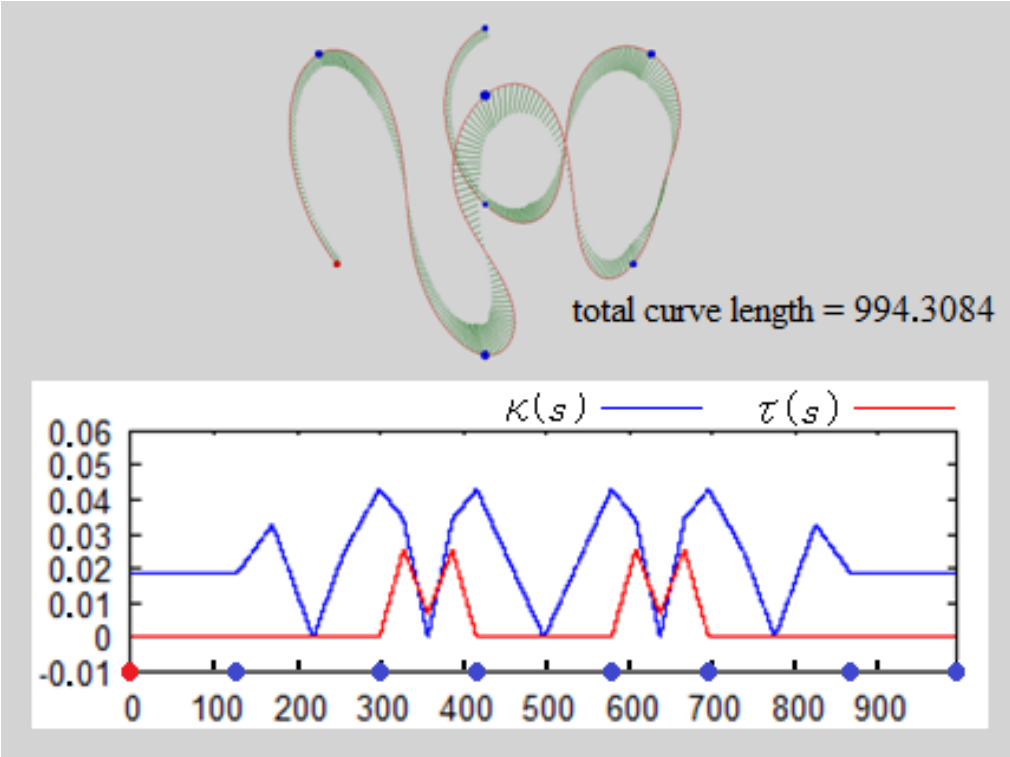
数値解析

2023年度前期　第12週 [7月6日]

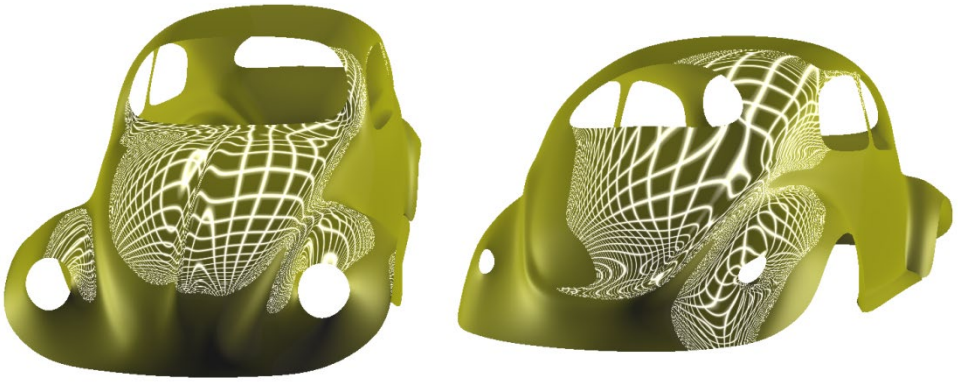


静岡大学
工学研究科機械工学専攻
ロボット・計測情報分野
創造科学技術大学院
情報科学専攻

三浦 憲二郎

講義アウトライン [7月6日]

- 関数近似と補間
  - 最小2乗近似による関数近似（復習）
  - ラグランジュ補間
  - 形状処理工学の基礎
    - 点列からの曲線の生成



関数近似　p.116（復習）

•複雑な関数を簡単な関数で近似する　　関数近似

•閉区間[a,b]で定義された関数f(x)を$g(x)=\Sigma a_i \phi_i(x)$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，

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

関数の差のノルムの二乗

$$\begin{aligned} ||f-g||_2^2 &:= \int_a^b \{f(x)-g(x)\}^2 dx \\ &= \int_a^b \{f(x)-\sum_{i=0}^{n-1} a_i\phi_i(x)\}^2 dx \end{aligned}$$

が最小になるように係数　a_i　を定める．

定理6.1　p.117（復習）

- 関数系　ϕ_i が1次独立であれば，(6.1)のノルム$\|\cdot\|_2^2$を最小にする係数は
  連立1次方程式

$$\begin{bmatrix} (\varphi_0,\varphi_0) & \cdots & (\varphi_0,\varphi_{n-1}) \\ \vdots & \ddots & \vdots \\ (\varphi_{n-1},\varphi_0) & \cdots & (\varphi_{n-1},\varphi_{n-1}) \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} (f,\varphi_0) \\ \vdots \\ (f,\varphi_{n-1}) \end{bmatrix}$$

の解として一意に決定される．ここで，[a, b] 上の連続関数f(x)とg(x)に対して

$$(f,g) = \int_a^b f(x)g(x)dx$$

定理6.1　p.117 (復習)

•(証明)　(6.1)より

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

が最小になるには極値の必要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(6.2)を解くことと同値.

グラム行列式　p.117

•要素 $x_0, x_1, \cdots, x_{n-1}$ が線形独立であることと，以下のグラム行列式が0でないことと同値．（6.2）はただ1つの解を持つ．

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

定理6.2　$\phi_i = x^i$ (i=0,1,2,…)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，

$$a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n = 0$$

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．

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_1 x を[0,1]で求めよ.

•$\phi_0(x)=1$, $\phi_1(x)=x$ とすると

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

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

したがって, a_0 = −1/6, a_1 = 1.　よって　y=x−1/6.

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), … ,(x_{m-1},y_{m-1})を通る関数f(x)を

$g(x)=\Sigma a_i\ \phi_i(x)$によって近似することを考える.

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

が最小になるようにa_0, a_1,…,a_{n-1}を決定する.

連立1次方程式

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

の解である.

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

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

最小2乗近似:プログラム:p.120

```
#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;
}
```

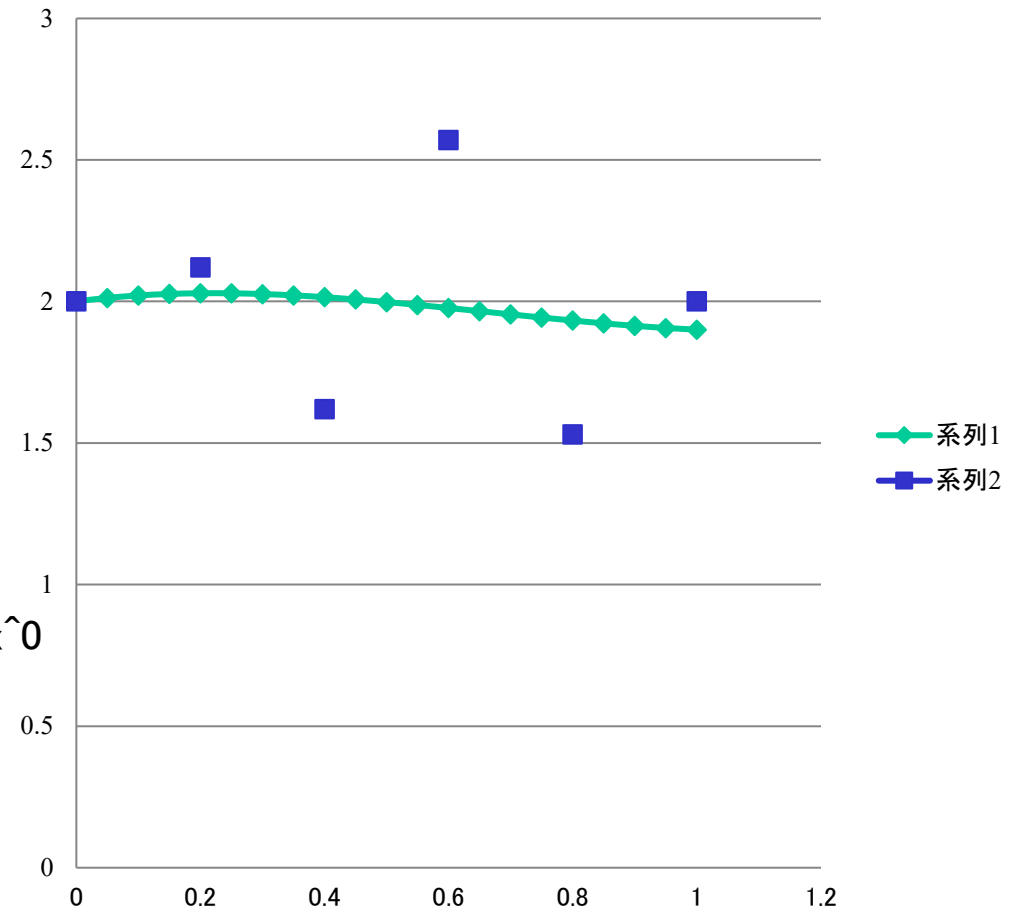
```
void least_square( double x[M], double y[M], FILE *fout )
{
  double a[N+1], p[N+1][N+1];
  int i, j, k;
  /* 右辺ベクトルの作成 */
  for(i=0; i <= N; i++)   {
    a[i]=0.0;
    for( j = 0; j < M; j++)   {
      a[i] +=  y[j]*pow(x[j],(double)i) ;
    }
  }
  /* 係数行列の作成 */
  for( i = 0; i <= N; i++)   {
    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 */
  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");
}
```

最小2乗近似：実行結果　p.123

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



ラグランジュ補間 p.124

•点(x_0,f_0), (x_1,f_1), … , (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補間，内挿 (interpolation)するという．

$f_k = f(x_k)$, $k=0,1,\cdots,n$ が与えられたとき，等式 $P(x_k) = f_k$, $k=0,1,\cdots,n$ を満たす多項式 $P(x)$ を $f(x)$ の補間多項式という．

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．

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

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

Vの行列式：バンデルモンド(Vandermonde)の行列，　解が1つに定まる．

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)

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

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

基本多項式 $l_i(x)$ の 点x_k（$0 \leqq k \leqq n$）での値は

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

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

ラグランジュ補間：プログラム　p.125

```
#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;
}
```

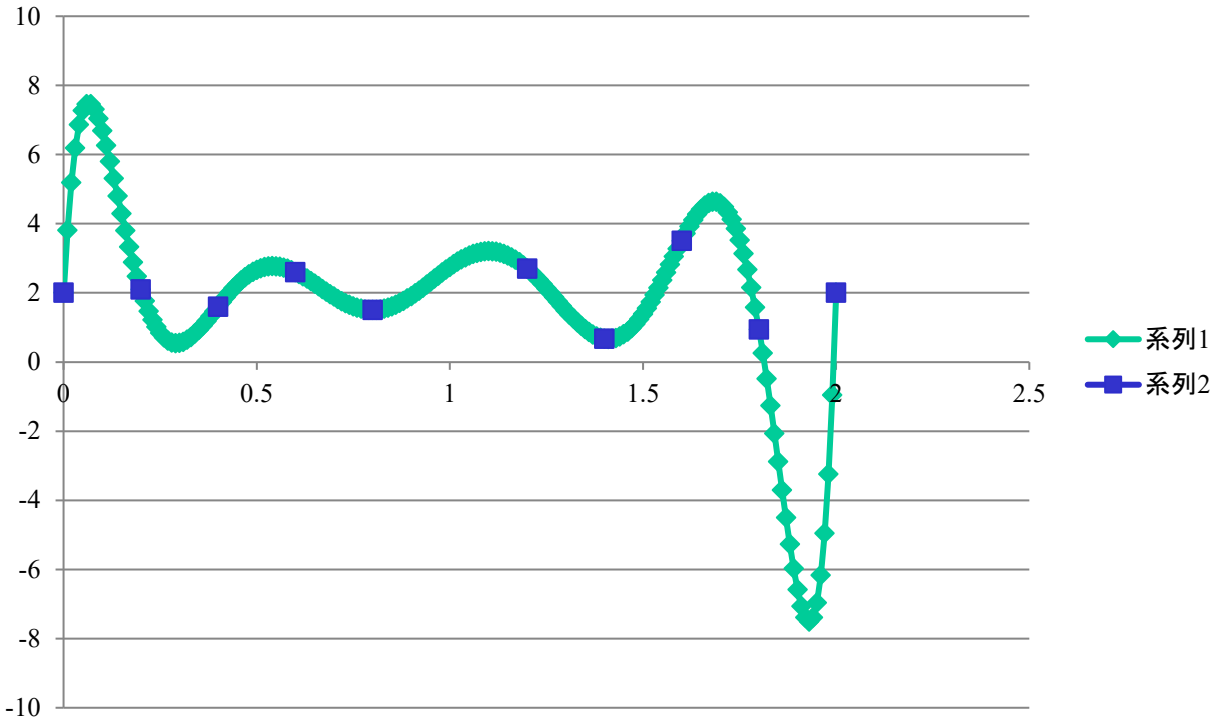
```
/* ラグランジュ補間 */
double lagrange( double x[N+1], double y[N+1], double xi )
{
  int i, k;
  double pn = 0.0, li;

  /* 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]);
  }
}
```

ラグランジュ補間：実行結果

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



関数近似　確認問題

1.最小2 乗法を用いて，3 点(0,-1), (1, 1), (2, 7) を近似する直線を求めよ.

2.ラグランジュ補間を用いて(0,-1), (1,1), (2, 7) を通る2 次式を求めよ.

3. 2 次関数$y = 2x^2 - 1$ を$0<=x <= 2$ の範囲で最良に近似する1 次関数$y=a_0 + a_1 x$を求めよ.

関数近似　確認問題 解答

1.最小2 乗法を用いて，3 点(1,-1), (2, 1), (3, 7) を近似する直線を求めよ.

$$y = 4x - \frac{5}{3}$$

2.ラグランジュ補間を用いて(0,-1), (1,1), (2, 7) を通る2 次式を求めよ.

$$y = 2x^2 - 1$$

3. 2 次関数$y = 2x^2 - 1$ を$0 <= x <= 2$ の範囲で最良に近似する1 次関数$y = a_0 + a_1 x$を求めよ.

$$y = 4x - \frac{7}{3}$$

形状処理工学の基礎

•形状処理工学 （Computer Aided Geometric Design）

かたちをコンピュータで処理する工学

・表現（データ構造），表示，変形

・リバースエンジニアリング

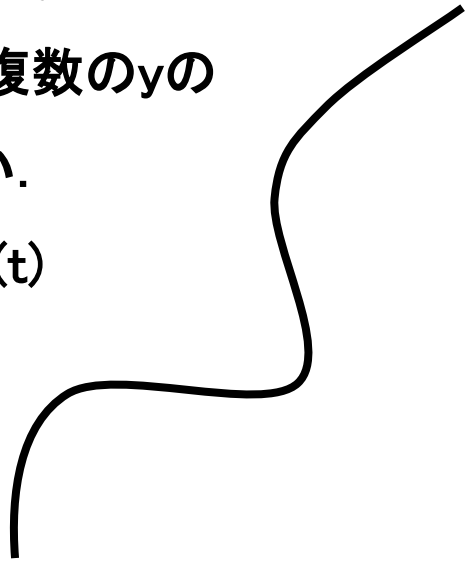
点群から曲面を再構築する.

•ここでは，点群の近似する曲線を生成することを考える.

y=f(x) の形式では，円のように，同じxの値に複数のyの値が対応する関数（多値関数）を表現できない.

そこで，パラメータ t を導入して曲線 C(t) を以下のように定義する.

$$C(t)=(x(t), y(t))$$



定理6.4の拡張

•m組の与えられた点$(x_0,y_0), (x_1,y_1), \cdots, (x_{m-1},y_{m-1})$とそれらのデータに対応するパラメータ値$t_0,t_1,\cdots,t_{m-1}$に対して，それらの点列を内挿する関数C(t)=(x(t),y(t))を$(g_x(t),g_y(t))=(\Sigma a_i\ \phi_i(t),\ \Sigma b_i\ \phi_i(t))$によって近似することを考える.

$$F(a_0,a_1,\cdots,a_{n-1},b_0,b_1,\cdots,b_{n-1}) := ||f-g||_2^2 = \sum_{k=0}^{m-1}\{(x_k-\sum_{j=0}^{n-1}a_j\phi_j(t_k))^2+(y_k-\sum_{j=0}^{n-1}b_j\phi_j(t_k))^2\}$$

が最小になるように$a_0, a_1,\cdots,a_{n-1}$および$b_0, b_1,\cdots,b_{n-1}$決定する.

2組の連立1次方程式

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

$$\begin{bmatrix}(\phi_0,\phi_0) & (\phi_0,\phi_1) & \cdots & (\phi_0,\phi_{n-1})\\ (\phi_1,\phi_0) & (\phi_1,\phi_1) & \cdots & (\phi_1,\phi_{n-1})\\ \vdots & \vdots & \ddots & \vdots\\ (\phi_{n-1},\phi_0) & (\phi_{n-1},\phi_1) & \cdots & (\phi_{n-1},\phi_{n-1})\end{bmatrix}\begin{bmatrix}b_0\\b_1\\\vdots\\b_{n-1}\end{bmatrix}=\begin{bmatrix}(\boldsymbol{y},\phi_0)\\(\boldsymbol{y},\phi_1)\\\vdots\\(\boldsymbol{y},\phi_{n-1})\end{bmatrix}$$

の解である.

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

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

点列の補間:プログラム

```
#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, FILE *fin, FILE *fout);
/*  行列の領域確保 */
/* 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乗近似呼び出し */
void least_square_coef( double x[M], double y[M], double a[N+1],  FILE *fout );
/* 最小2乗近似 */
void least_square( double x[M], double y[M], FILE *fout );
/* 曲線の生成 */
void curve( double x[M], double y[M], double t[M], FILE *fout);
/* 曲線上の点列データ出力 */
void curve_shape( double a0[N+1], double b0[N+1], FILE *fp1);
double a[N+1], b[N+1]; /* グローバル変数 */
int main(void){
  FILE *fin, *fout, *fout1;
  double x[M], y[M], t[M];
  /* ファイルのオープン */
  if ( (fin = fopen( "curve_fitting.dat", "r")) == NULL ){
     printf("ファイルが見つかりません : curve_fitting.dat ¥n"); exit(1);
  }
  if( (fout = fopen( "output_curve.dat", "w")) == NULL ){
    printf("ファイルが作成できません : output_curve.dat ¥n"); exit(1);
   }
  if( (fout1 = fopen( "output_shape.dat", "w")) == NULL ){
    printf("ファイルが作成できません : output_shape.dat ¥n");exit(1);
   }
  input_vector2( x, 'x', fin, fout );    /* ベクトルxの入出力 */
  input_vector2( y, 'y', fin, fout );    /* ベクトルyの入出力 */
  input_vector2( t, 't', fin, fout );    /* パラメータtの入出力 */
  curve( x, y, t, fout);  /* 最小2乗近似 */
  curve_shape( a, b, fout1);
  fclose(fin); fclose(fout);  /* ファイルのクローズ */
  return 0;
}
```

```
void curve( double x[M], double y[M], double t[M], FILE *fout)
{
          least_square_coef( t, x, a, fout);
          least_square_coef( t, y, b, fout);
}

void least_square_coef( double x[M], double y[M], double a[N+1],
FILE *fout )
{
 double 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 );
```

点列の補間：プログラム その2

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

/* 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 )
{
  int i, j, k, ip;
  double alpha, tmp;
  double amax, eps=pow(2.0, -50.0); /* eps = 2^{-50}とする */
  for( k = 0; k < n-1; k++)
  {
    /* ピボットの選択 */
    amax = fabs(a[k][k]); ip = k;
    for( i = k+1; i < n; i++){
      if ( fabs(a[i][k]) > amax ){
        amax = fabs(a[i][k]); ip = i;
      }
    }
    /* 正則性の判定 */
    if ( amax < eps ) printf("入力した行列は正則ではない!!¥n");
    /* 行交換 */
    if ( ip != k){
      for( j = k; j < n; j++){
        tmp = a[k][j]; a[k][j]=a[ip][j]; a[ip][j]=tmp;
      }
        tmp = b[k] ; b[k]=b[ip]; b[ip]=tmp;
    }
    /* 前進消去 */
    for( i = k+1; i < n; i++){
      alpha = - a[i][k]/a[k][k];
      for( j = k+1; j < n; j++){
        a[i][j] = a[i][j] + alpha * a[k][j];
      }
      b[i] = b[i] + alpha * b[k];
    }
  }

  /* 後退代入 */
  b[n-1] = b[n-1]/a[n-1][n-1];
  for( k = n-2; k >= 0; k--)
  {
    tmp = b[k];
    for( j = k+1; j <= n; j++)
    {
      tmp = tmp - a[k][j] * b[j];
    }
    b[k] = tmp/a[k][k];
  }
}

void curve_shape( double a0[N+1], double b0[N+1], FILE *fp1)
{
        int i,j;
        double dt0=0.05;
        double x0,y0,t0=0.,t1;
        for(i=0;i<=20;++i){
                x0=a0[0];
                y0=b0[0];
                t0+=dt0;
                t1=t0;
                for(j=1;j<=N;++j){
                        x0+=a0[j]*t1;
                        y0+=b0[j]*t1;
                        t1*=t0;
                }
                fprintf(fp1,"%lf¥t%lf¥n",x0,y0);
        }
}
```

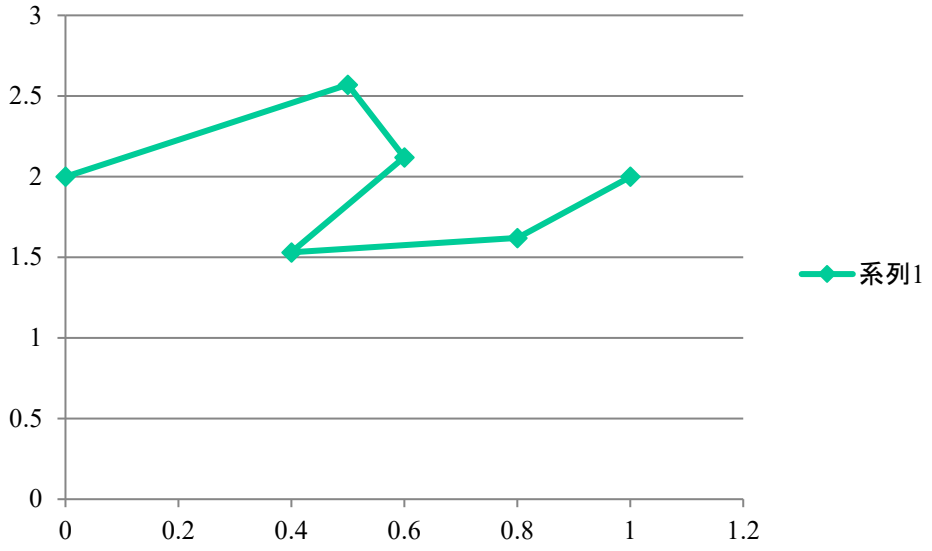
ラグランジュ補間：実行結果

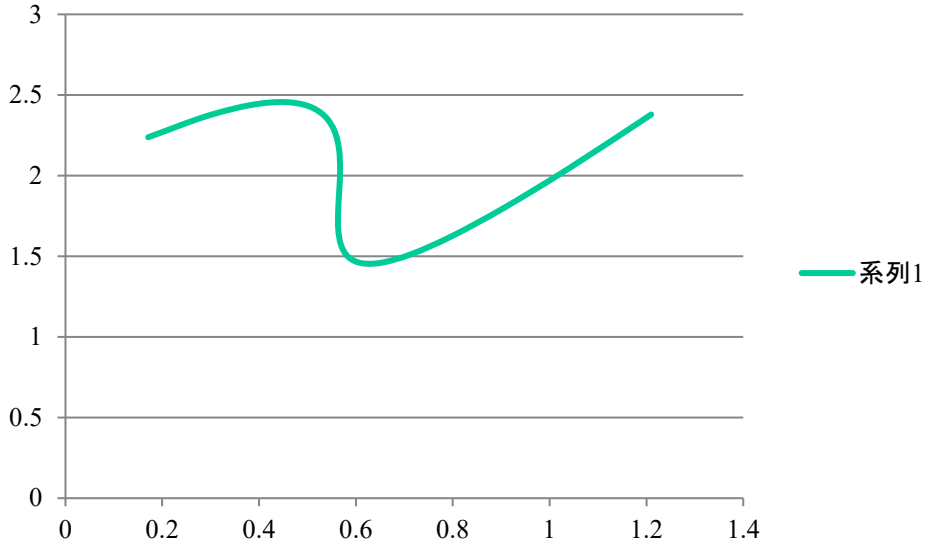
curve_fitting.dat

0.0 0.5 0.6 0.4 0.8 1.0

2.0 2.57 2.12 1.53 1.62 2.0

0.0 0.2 0.4 0.6 0.8 1.0





連絡先

- 電子メイルアドレス　miura.kenjiro@shizuoka.ac.jp
- ホームページ

  https://mc2-lab.com/
- 電話・ファックス　053-478-1074
- 授業用ホームページ　https://mc2-lab.com/lecture.html#A

まとめ

- 関数近似と補間
  - 最小2乗近似による関数近似
  - ラグランジュ補間
  - 形状処理工学の基礎
    - 点列からの曲線の生成

- 受講、お疲れ様でした。