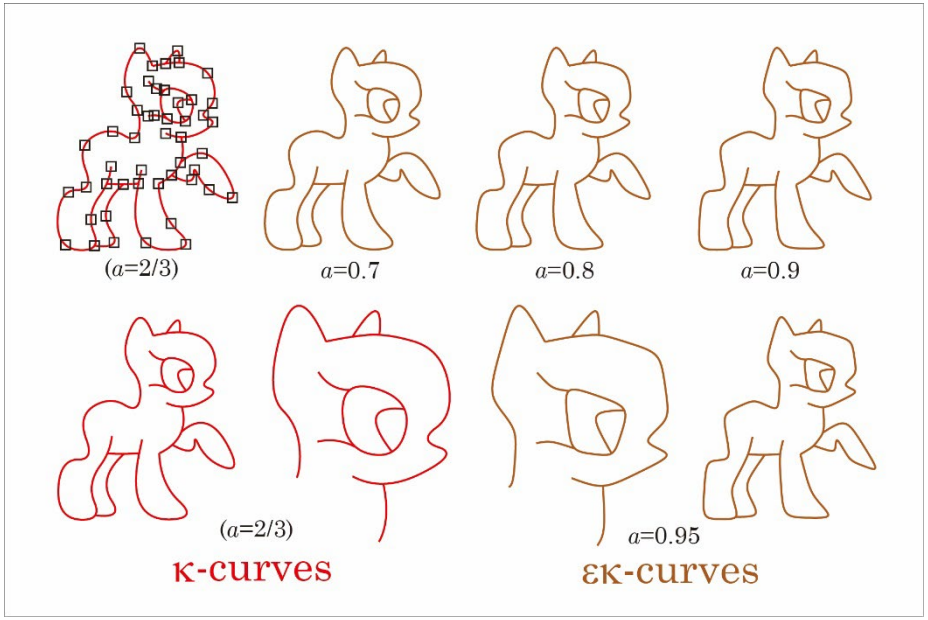
数値解析

2023年度前期　第9週 [6月15日]



静岡大学大学院
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 [6月15日]

- 連立1次方程式の反復解法
  - 密行列と疎行列
  - 反復法の原理：縮小写像の原理
  - ヤコビ法
  - ガウス・ザイデル法
  - データ入出力　2重のポインターの用法

連立一次方程式の反復解法 p.93

•連立1次方程式

$$Ax = b$$

•密(dense)行列と疎(sparse)行列

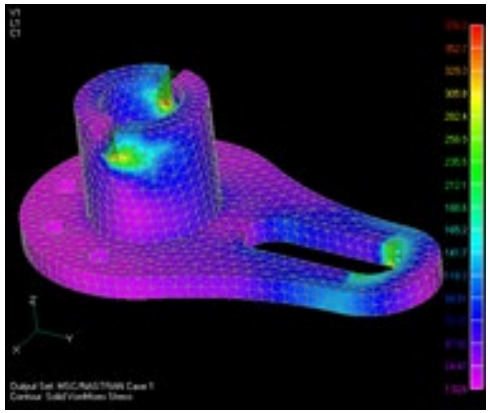
密行列：行列の要素のすべて，あるいはほとんどゼロでない行列

疎行列：行列の要素のほとんどがゼロである行列

•疎行列

•反復解法を用いるほうが速い．

•ヤコビ法とガウスザイデル法



反復法の原理 p.93

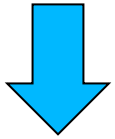
連立1次方程式

$$Ax = b \quad (5.1)$$

を同値な方程式

$$x = \phi(x) = Mx + Nb$$

に変形し，反復法により解を求める．Mを反復行列という．



縮小写像の原理

縮小写像の原理　定理5.1　p.94

R^n 上で定義された関数$g(x)$ が

(1) $x \in R^n \rightarrow g(x) \in R^n$

(2) $x, y \in R^n \rightarrow ||g(x) - g(y)|| \leq L||x - y|| \in R^n$

(3) $0 \leq L < 1$

を満たすとき，方程式

$$x = g(x)$$

の解$x = \alpha$はR^nにおいてただ1つ存在する．そして，$x^{(0)}$を初期値とする反復に

$$x^{(k+1)} = g(x^{(k)})$$

によって，生成される列$\{x^{(k)}\}$は$k \rightarrow \infty$のときαに収束する．

縮小写像の原理　定理5.2　p.94

あるノルム$\|\cdot\|$について$||\mathrm{M}|| < 1$ならば反復法

$$x^{(k+1)} = \phi\left(x^{(k)}\right) = Mx^{(k)} + Nb$$

によって作られる列$\{x^{(k)}\}$は(5.1)の解xに収束する

(証明)

任意の$x, y \in R^n$に対して,

$$||\phi(x) - \phi(y)|| = ||Mx - My|| \leq ||M|| \cdot ||x - y|| \leq ||x - y||$$

が成り立つので, $\phi(x)$は定理5.1の仮定を満たす. よって, $\{x^{(k)}\}$は解xに収束する.

行列Aの分解　p.95

$$A = D + L + U$$

$$D = \begin{bmatrix} a_{11} & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & a_{nn} \end{bmatrix}$$

$$L = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & 0 \end{bmatrix}$$

$$U = \begin{bmatrix} 0 & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$$

ヤコビ法　p.95

$$M = -D^{-1}(L+U), N = D^{-1}$$

とおいた反復法

$$x^{(k+1)} = D^{-1}\{b - (L+U)x^{(k)}\}$$

ここで，　D^{-1}の存在を仮定している.

アルゴリズム

Input $A, b, x^{(0)}, \varepsilon, kmax$

$k \leftarrow 0$

Do

　For $i = 1, 2, \ldots, n$

　　$x_i^{(k+1)} \leftarrow \frac{1}{a_{11}}(b_i - \sum_{j \neq i}^{n} a_{ij} x_j^{(k)})$

　end for

ヤコビ法　p.95

アルゴリズム

Input $A, b, x^{(0)}, \varepsilon, kmax$

$k \leftarrow 0$

Do

　For $i = 1, 2, \ldots, n$

$$x^{(k+1)} \leftarrow \frac{1}{a_{11}} \left(b_i - \sum_{j \neq i}^{n} a_{ij} x_j^{(k)}\right)$$

　end for

　$k \leftarrow k + 1$

while $\left(\left\|x^{(k+1)} - x^{(k+1)}\right\| > \varepsilon \text{ and } k < k_{max}\right)$

If $k = k_{max}$ “失敗” else Output $x^{(k+1)}$ end if

ヤコビ法:プログラム その1

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define N       10                  /* N元方程式 */
#define EPS     pow(10.0, -8.0)     /* epsilonの設定 */
#define KMAX    100

/* 行列の入力 */
void input_matrix(double **a, char c, FILE* fin, FILE* fout);
/* ベクトルの入力 */
void input_vector(double *b, char c, FILE* fin, FILE* fout);
/* 行列の領域確保 */
double **dmatrix(int nr1, int nr2, int nl1, int nl2);
/* 行列の領域解法 */
void free_dmatrix(double** a, int nr1, int nr2, int nl1, int
nl2);
/* ベクトル領域の確保 */
double* dvector(int i, int j);
/* ベクトルの領域解法 */
void free_dvector(double* a, int i);
/* 比較関数 */
int double_comp(const void *s1, const void *s2);
/* 最大値ノルムの計算 a[m...n] */
double vector_norm_max(double *a, int m, int n);
/* ヤコビ法 */
double* jacobi_lin(double** a, double* b, double* x);
```

ヤコビ法:プログラム その2

```
int main(void)
{
    FILE* fin, * fout;
    double** a, * b, * x;
    int i;
    /* 行列およびベクトルの領域確保 */
    a = dmatrix(1, N, 1, N); /* 行列 a[1. . .N] [1. . .N] */
    b = dvector(1, N); /* b[1...N] */
    x = dvector(1, N); /* x[1...N] */
    /* ファイルのオープン */
    if ((fin = fopen("input_sp.dat", "r")) == NULL){
        printf("ファイルが見つかりません : input_sp.dat ¥n");
        exit(1);
    }
    if ((fout = fopen("output_sp.dat", "w")) == NULL){
        printf("ファイルが作成できません : output_sp.dat ¥n");
        exit(1);
    }
    input_matrix(a, 'A', fin, fout); /* 行列Aの入出力 */
    input_vector(b, 'b', fin, fout); /* ベクトルbの入出力 */
    input_vector(x, 'x', fin, fout); /* 初期ベクトルx0の入出力 */
    x = jacobi_lin(a, b, x);
    /* 結果の出力 */
    fprintf(fout, "Ax=b の解は次の通りです¥n");
    for (i = 1; i <= N; i++){
        fprintf(fout, "%f¥n", x[i]);
    }
    fclose(fin); fclose(fout); /* ファイルのクローズ */
    /* 領域の解法 */
    free_dmatrix(a, 1, N, 1, N); free_dvector(b, 1);
    free_dvector(x, 1);
    return 0;
}
```

ヤコビ法:プログラム その3

```
/* ヤコビ法 */
double* jacobi_lin(double** a, double* b, double* x)
{
    double eps, *xn;
    int i, j, k = 0;
    xn = dvector(1, N); /* xn[1...N] */
    /* x <- x_k, xn <- x_{k+1} */
    do{
        for (i = 1; i <= N; i++){
            xn[i] = b[i];
            for (j = 1; j <= N; j++){
                xn[i] -= a[i][j] * x[j];
            }
            xn[i] += a[i][i] * x[i];
            xn[i] /= a[i][i];
        }
        for (i = 1; i <= N; i++) x[i] = xn[i] - x[i];
        eps = vector_norm_max(x, 1, N); /* 最大値ノルムの計算 */
        for (i = 1; i <= N; i++) x[i] = xn[i]; /* 値を更新 */
        k++;
    } while (eps > EPS && k < KMAX);
    free_dvector(xn, 1); /* 領域の解法 */
    if (k == KMAX)
    {
        printf("答えが見つかりませんでした¥n");
        exit(1);
    }
    else
    {
        printf("反復回数は%dです¥n", k); /* 反復回数を画面に表示 */
        return x;
    }
}
```

ガウス･ザイデル法　p.100

$$M = -(D + L)^{-1}U, N = (D + L)^{-1}$$

とおいた反復法

$$x^{(k+1)} = D^{-1}\{b - Lx^{(k+1)} - Ux^{(k)}\}$$

ここで，　D^{-1}の存在を仮定している.

アルゴリズム

Input $A, b, x^{(0)}, \varepsilon, kmax$

$k \leftarrow 0$

Do

　For $i = 1, 2, \ldots, n$

$$x_i^{(k+1)} \leftarrow \frac{1}{a_{11}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j\neq i}^{n} a_{ij}x_j^{(k)}\right)$$

　end for

ガウスザイデル法:プログラム p.101

```
/* ガウス･ザイデル法 */
double* gauss_seidel(double** a, double* b, double* x)
{
     double eps, * xo, s, t;
     int i, j, k = 0;
     xo = dvector(1, N); /* xo[1...N] */
     do{
          /* xo <- x_k, x <- x_{k+1} */
          for (i = 1; i <= N; i++) xo[i] = x[i]; /* x_kにx_(k+1)を代入 */
          /* i=1の処理 */
          t = 0.0;
          for (j = 2; j <= N; j++) t += a[1][j] * xo[j];
          x[1] = (b[1] - t) / a[1][1];
          /* i=2,3, . . .N の処理 */
          for (i = 2; i <= N; i++){
               s = 0.0; t = 0.0;
               for (j = 1; j < i; j++) s += a[i][j] * x[j];      /* i-1列までの和 */
               for (j = i+1; j <= N; j++) t += a[i][j] * x[j];  /* i+1列以降の和 */
               x[i] = (b[i] - s - t) / a[i][i];
          }
          for (i = 1; i <= N; i++) xo[i] = xo[i] - x[i];
          eps = vector_norm_max(xo, 1, N);
          k++;
     } while (eps > EPS && k < KMAX);
     free_dvector(xo, 1); /* */
     if (k == KMAX){
          printf("答えはがみつかりませんでした¥n");
          exit(1);
     }
     else{
          printf("反復回数は%dです¥n", k); /* 反復回数を画面に表示 */
          return x;
     }
}
```

データ入出力　p.19 (「明快入門C」p.161)

プログラム2.3

```
#include <stdio.h>
#include <stdlib.h>
#define N 4
void input_matrix(double **a,char c,FILE* fin,FILE* fout);
void input_vector(double *b,char c,FILE* fin,FILE* fout);
double **dmatrix(double **a,int nr1,int nr2,int nl1,int nl2);
Void free_dmatrix(double **a,int nr1,int nr2,int nl1,int nl2);
double *dvector(double *a, int i);
Void free_dvector(double *a, int i);
int main(void)
{
    FILE *fin,*fout;
    double **a,*b;
    a = dmatrix(1, N, 1, N); /* A[1…N][1…N] */
    b = dvector(1,N); /* b[1…N] */
```

データ入出力　p.19 (「明快入門C」p.161)

プログラム2.3 続き

```
if((fin=fopen("input.dat","r"))==NULL){
    printf("ファイルは見つかりません:input.dat ¥n");
    exit(1);
}
if((fout=fopen("output.dat","w"))==NULL){
    printf("ファイルが作成できません:output.dat ¥n");
    exit(1);
}
```

データ入出力　p.19 (「明快入門C」p.161)

プログラム2.3 続き

```
    input_matrix(a,'A',fin,fout); /* 行列Aの入出力 */
    input_vector(b,'b',fin,fout); /* ベクトルbの入出力 */

    fclose(fin); fclose(fout);
}
void input_matrix(double **a,char c,FILE* fin,FILE* fout){
    int i,j;
    fprintf(fout,"行列%cは次の通りです¥n",c);
    for(i=0;i<N;++i){
        for(j=0;j<N;++j){
            fscanf(fin,"%lf",&(a[i][j]));
            fprintf(fout,"%5.2f¥t",a[i][j]);
        }
        fprintf(fout,"¥n");
    }
}
```

データ入出力　p.19 (「明快入門C」p.161)

プログラム2.3 続き

```
void input_vector(double *b,char c,FILE* fin,FILE* fout){
    int i;
    fprintf(fout,"ベクトル%cは次の通りです¥n",c);
    for(i=0;i<N;++i){
        fscanf(fin,"%lf",&(b[i]));
        fprintf(fout,"%5.2f¥t",b[i]);
        fprintf(fout,"¥n");
    }
}
```

input.datの内容

```
1.0   2.0   1.0   1.0
4.0   5.0 -2.0   4.0
4.0   3.0 -3.0   1.0
2.0   1.0   1.0   3.0
-1.0 -7.0 -12.0 2.0
```

データ入出力　p.19 (「明快入門C」p.161)

プログラム2.3改 続き

output.datの内容

行列Aは次の通りです.

```
1.0   2.0   1.0   1.0
4.0   5.0  -2.0   4.0
4.0   3.0  -3.0   1.0
2.0   1.0   1.0   3.0
```

ベクトルbは次の通りです.

```
-1.0
-7.0
-12.0
 2.0
```

まとめ

- 連立1次方程式の反復解法
  - 密行列と疎行列
  - 反復法の原理：縮小写像の原理
  - ヤコビ法
  - ガウス・ザイデル法
  - データ入出力　2重のポインターの用法