Cプログラミング

第5回
非線形方程式の解法(1)

青柳滋己
aoyagi@aoni.waseda.jp

本日の講義内容

- UNIXの復習[1]
- 構造体（復習）
- 前回の復習
- 非線形方程式の解法(1)
  - 2分法
- 演習

UNIXのcshの機能[参考]

- csh(C-シェル)
  - cshとは…ユーザの入力を理解して各種コマンドを実行するコマンドインタプリタ．GNOME端末を起動すると裏で動いている．
    - bash(B-シェル)というのもある．

      cf. Windowsのコマンドプロンプト
  - cshの機能
    - ヒストリー…(過去の入力を呼び出す)
    - コマンドラインエディタ…入力行の編集
      - 矢印キー
      - Emacsキーバインド: C-f, C-b, C-a, C-e, C-p, C-n
    - ファイル名展開(Tabキー)…ファイル名を途中まで入力してTabキーを打つと，残りの部分を展開してくれる．

cshのパターンマッチ[参考]

- cshのパターンマッチ規則
  - * 任意の文字列
  - ? 任意の一文字
  - [*ccc*] *ccc*の中のどれか一文字

```
% ls
hello  hello.c  samp1  samp1.c  samp2.c  samp3.c  samp10.c
test.c work/
% ls samp?.c     /* samp○.cというタイプのファイルを表示 */
samp1.c  samp2.c  samp3.c
% ls *.c         /* 拡張子が.cであるファイルを表示 */
hello.c  samp1.c  samp2.c  samp3.c  samp10.c  test.c
% ls [a-n]*      /* ファイルの先頭がaからnまでの文字で始まるファイル */
hello  hello.c
```

リダイレクト機能(csh)

入出力の切り替え(リダイレクト)

実行するコマンドやプログラムの画面への出力をファイル化したり，キーボード入力の代わりにファイルに書かれた文字を入力する機能.

- コマンド ＞ file名

  標準出力(画面)に出力される内容を file名 のファイルに書き込む. file名のファイルが存在する場合には書き換えられる. 画面には何も出力されなくなる.

- コマンド ＞＞ file名

  ">" と同じだが，file名のファイルが存在する時はファイルは消されず，ファイルの最後に追加される.

- コマンド ＜ file名

  キーボードからの入力の代わりに，file3があたかもキーボードから入力されたかのように振舞う.

注：Windows(コマンドプロンプト)でも同様の機能がある.

リダイレクトの例(1/2)

- 標準出力をファイルへ

```
% ls                              /* 赤字部分がユーザの手入力 */
hello hello.c
% ./hello
Hello, world.                     /* 通常は画面に出力される．  */
% ./hello > output.txt            /* 画面には何も出力されなくなる．  */
% ls
hello  hello.c  output.txt        /* output.txtというファイルができている．*/
% cat output.txt                  /* cat はファイルの内容を画面に表示する．*/
Hello, world.
%
```

リダイレクトの例(2/2)

ファイルを標準入力へ

test.c

```
#include <stdio.h>

int main(void) {
  int num1, num2;

  printf("num1? : ");
  scanf("%d", &num1);
  printf("num2? : ");
  scanf("%d", &num2);
  printf("Answer: %d¥n", num1 * num2);
  return 0;
}
```

```
% ls
input.txt  test.c
% gcc -o test test.c
% ./test
num1? : 2030    /* キーボード入力 */
num2? : 56321   /* キーボード入力 */
Answer: 114331630
% cat input.txt
543
198765
% ./test < input.txt
num1? : num2? : Answer: 107929395
% ./test < input.txt > output.txt
% cat output.txt
num1? : num2? : Answer: 107929395
%
```

リダイレクトの応用

- エディタを使わずに小さなファイルを作成できる
- 終了するには行頭でCTRL-dを入力する.
- Windows(コマンドプロンプト)の場合はtypeコマンドを使い，"type con > ファイル名"とする．また，終了は行頭でCTRL-z，続けてEnterで終了．
- 前の行に戻って編集したりできないので注意.

```
% ls      /* カレントディレクトリには何もない */
% cat > test.txt
123             /* ここからの3行は手で打ち込む */
redirect test sample
^D      /* 終了は行頭でCTRL-d. この行は実際は出力されない */
% ls
test.txt  /* test.txtというファイルができている */
% cat > test.txt  /* test.txtを表示 */
123
redirect test sample
%
```

```
C:¥WINDOWS¥Temp> type con > test.txt
123
test sample
^Z              /* 行の先頭で CTRL-z , 続けてEnter */
C:¥WINDOWS¥Temp> dir  /* dirはUNIXのlsに相当 */
test.txt
C:¥WINDOWS¥Temp> type test.txt   /* test.txtを表示 */
123
test sample

C:¥WINDOWS¥Temp>
```

構造体(復習)

- 複数の変数をまとめて組にして扱える.
- 複数の属性を持つものを扱うときに使うと良い.
- まず最初に構造体の宣言を行う. 宣言はその構造体のテンプレートを定義する. 変数を続けて宣言してもよい.

```
struct タグ名
{
  第1メンバの宣言;
  第2メンバの宣言;
  第3メンバの宣言;
   .....
} 変数名;
```

```
.....
struct book
{
  char title[256];
  char author[256];
  int pages;
};

int main(void)
{
  struct book a1:

  strcpy(a1.title, "C nyuumon");
  strcpy(a1.author, "H. Hayashi");
  a1.pages = 322;

  .....
}
```

構造体変数(復習)

- 構造体変数を使うには，使用宣言する．

```
struct タグ名 変数名;
```

- 構造体のメンバにアクセスするには，"." を使う．
- タグ名を使わずに変数宣言することもできる．

```
struct
{
  char title[256];
  char author[256];
  int pages;
} a1;
```

- =で代入が可能．各メンバがそのままコピーされる．
- メンバに他の構造体を持つことも可能．

```
.....
struct book
{
  char title[256];
  char author[256];
  int pages;
};

int main(void)
{
  struct book a1, b1:

  .....
  printf("title=%s¥n", a1.title);
  printf("author=%s¥n", a1.author);
  printf("pages=%d¥n", a1.pages);
  b1 = a1;
}
```

構造体ポインタ(復習)

- 構造体変数を指す構造体ポインタは他のポインタ変数と同様に宣言する.
- 構造体ポインタの指す構造体のメンバを参照するには，アロー演算子"->"を使う.

注: アロー演算子を使わない場合は演算子の優先順位に注意．例えば

```
b->pages = 100;
```

は

✕ `*b.pages = 100;`

(∵ `*(b.pages) = 100;` と解釈される)

○ `(*b).pages = 100;`

```
.....
struct book
{
  char title[256];
  char author[256];
  int pages;
};

int main(void)
{
  struct book a1, *b:

  b=&a1;
  printf("title=%s¥n", b->title);
  printf("author=%s¥n", b->author);
  printf("pages=%d¥n", b->pages);
  ....
}
```

構造体と配列(復習)

- 構造体の配列は通常の変数の宣言と同じように宣言する.
- 構造体メンバへは"."でアクセス可能. ポインタを使って"->"を使うこともできる.

```
struct book
{
  char title[256];
  char author[256];
  int pages;
};

int main(void)
{
  struct book a[3], *b;
  int i;

  a[0].pages=200; /* 初期化 */
  .....
  for (i=0; i<3; i++) {
    b=&a[i];
    printf("title=%s¥n", b->title);
    printf("author=%s¥n", b->author);
    printf("pages=%d¥n", b->pages);
  }
  .....
}
```

構造体を関数引数にする

- 構造体を引数に渡すことができる.
  - 構造体メンバの値は全部コピーされる.
  - 構造体配列の場合は普通の配列と同じくポインタが渡される.
  - 呼び出した関数内で値を変更したい場合は必ずポインタで渡すこと.

```
.....
void viewstrct(struct book b)
{
  b.pages=150;
  printf("title=%s¥n", b.title);
  printf("author=%s¥n", b.author);
  printf("pages=%d¥n", b.pages);
}

int main(void)
{
  struct book a1;
  .....
  viewstrct(a1);
  printf("title=%s¥n", a1.title);
  printf("author=%s¥n", a1.author);
  printf("pages=%d¥n", a1.pages);
  ...
}
```

typedef宣言

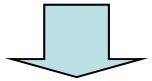
- 新しいデータ型の名前を作成する
- 新しい型を作るのではなく，"新しい型の名前"を作る

typedef　既存のデータ型　同義語名

- 例

```
typedef unsigned int UINT;

UINT num;
```



上は

```
unsigned int num;
```

と解釈される．

```
typedef struct
{
  char title[256];
  char author[256];
  int pages;
} BOOK;


BOOK b1,b2; /* 変数宣言 */
```

構造体変数の宣言にstruct…とつけなくてもよくなる．

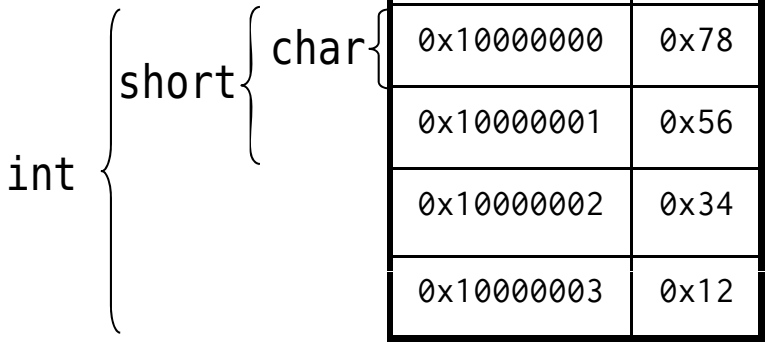
共用体(参考)

- 構造体と似たような記述法
- メモリを最大にとるメンバが用いるメモリ量が確保され，全メンバでそのメモリを共用する.

```
.....
union sampleun
{
  char cn;
  short sn;
  int num;
};

int main(void)
{
  union sampleun test1;

  test1.num=0x12345678;
  printf(“test1.num=%d¥n”, test1.num);
  printf("test1.sn=%x¥n", test1.sn);
  printf("test1.cn=%x¥n", test1.cn);
  .....
}
```



アドレス	内容
0x10000000	0x78
0x10000001	0x56
0x10000002	0x34
0x10000003	0x12

ピボット選択付きガウス消去法(1/3)(復習)

- ガウス消去法では対角成分が0となるものは解けない

$$\begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 3 \\ 3 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ -3 \end{pmatrix}$$

前進消去の際，対角成分a[k][k]は絶対値が最大となるものに行全体を交換する．上の例ではまず第1列の最大値は3．よって第3行と第1行を入れ替える．

$$\begin{pmatrix} 3 & 1 & 0 \\ 1 & 0 & 3 \\ 0 & 1 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -3 \\ 2 \\ 2 \end{pmatrix}$$

ピボット選択付きガウス消去法(2/3)(復習)

$$\begin{pmatrix} 3 & 1 & 0 \\ 1 & 0 & 3 \\ 0 & 1 & 2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -3 \\ 2 \\ 2 \end{pmatrix}$$

- 第1列について前進消去を行う

$$\begin{pmatrix} 3 & 1 & 0 \\ 0 & -\frac{1}{3} & 3 \\ 0 & 1 & 2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -3 \\ 3 \\ 2 \end{pmatrix}$$

- 第2列については第2行目以降について絶対値が最大なものを探して第2行目と入れ替える.

$$\begin{pmatrix} 3 & 1 & 0 \\ 0 & 1 & 2 \\ 0 & -\frac{1}{3} & 3 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -3 \\ 2 \\ 3 \end{pmatrix}$$

ピボット選択付きガウス消去法(3/3)(復習)

$$\begin{pmatrix} 3 & 1 & 0 \\ 0 & 1 & 2 \\ 0 & -\frac{1}{3} & 3 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -3 \\ 2 \\ 3 \end{pmatrix}$$

- 第2列について前進消去を行う

$$\begin{pmatrix} 3 & 1 & 0 \\ 0 & 1 & 2 \\ 0 & 0 & \frac{11}{3} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -3 \\ 2 \\ \frac{11}{3} \end{pmatrix}$$

- 以降は後退代入を行い，解を求める

ピボット選択付きガウス消去法

- 部分ピボット選択を入れる

```
for k=1,2,...,n-1
   amax=|a[k][k]|   ip=k
   for i=k+1,k+2,...,n
      if (|a[i][k]| > amax) then
         amax=|a[i][k]|
         ip=i
      end
   end
   if (ip≠k)  then
      for j=k,k+1,…,n
        swap(a[k][j], a[ip][j])
      end
      swap(b[k], b[ip])
   end
   /* ここで前進消去 */
end
/* ここで後退代入 */
```

前回の演習の解答例

kadai4_1.c

```
.......
int main(int argc, char *argv[])
{
  int i,j, N;
  FILE *fp;
  double **mat1,*mat2;

  if (argc != 2) {
    printf("Usage: %s <datafile>¥n", argv[0]);
    return 0;
  }
  if ((fp=fopen(argv[1],"r")) == NULL) {
    printf("cannot open %s¥n", argv[1]);
    exit(1);
  }
  fscanf(fp, "%d", &N); /* 次数読み込み */
  if ((mat1=(double **)malloc((N+1)
            *sizeof(double *))) == NULL) {
    printf("malloc error¥n");
    exit(1);
  } /* ポインタの配列分のメモリ確保 */
/* 右へ続く */
```

kadai4_1.c

```
for (i=1; i<=N; i++) { /* Aの1行分メモリ確保 */
  if ((mat1[i]=(double *)malloc((N+1)
                  *sizeof(double))) == NULL) {
    printf("malloc error¥n");
    exit(1);
  }
  for (j=1; j<=N; j++) {/* Aのデータ読み込み */
    fscanf(fp, "%lf", &mat1[i][j]);
  }
}
if ((mat2=(double *)malloc((N+1)
                *sizeof(double))) == NULL) {
  printf("malloc error¥n");
  exit(1);
}
for (i=1; i<=N; i++) { /* Bのデータ読み込み */
  fscanf(fp, "%lf", &mat2[i]);
}
gauss(mat1, mat2, N);
....... /* 以下省略 */
```

前回の演習の解答例

- mathライブラリを使う時にはmath.hをincludeする．また，コンパイル時に -lm オプションをつけること．

kadai4_2.c

```
.......
void gauss2(double **a, double *b, int N)
{
  int i,j,k, ip;
  double alpha, tmp;
  double amax;

  for (k=1; k<=N-1; k++) { /* 前進消去ループ開始 */
    amax=fabs(a[k][k]); ip=k;  /* ピボット選択 */
    for (i=k+1; i<=N; i++) { /* 最大値を探す */
      if (fabs(a[i][k])>amax) {
          amax=fabs(a[i][k]);
        ip=i;
      }
    }
    if (ip != k) { /* 行の入れ替え */
      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;
    }
/* 以下，前進消去，後退代入を行う */
```

前回の演習の解答例(3/3)

kadai5_4.c

```
#include <stdio.h>
#include <string.h>

#define MAXLINE 80

void upper(char *st)
{
  int i, len;

  len=strlen(st);
  for (i=0; i<len; i++)
    if ((st[i] >= 'a') && (st[i] <= 'z')) {
      st[i] -= 'a'- 'A';
    }
}
```

kadai4_3.c

```
void lower(char *st)
{
  int i, len;

  len=strlen(st);
  for (i=0; i<len; i++)
    if ((st[i] >= 'A') && (st[i] <= 'Z')) {
      st[i] += 'a'- 'A';
    }
}
```

ASCIIコード表(再掲)

- 文字は下表の数字として扱う

	0x00	0x10	0x20	0x30	0x40	0x50	0x60	0x70
0x0	¥0 (NUL)	(制御文字)	空白文字	0	@	P	`	p
0x1	(制御文字)	(制御文字)	!	1	A	Q	a	q
0x2	(制御文字)	(制御文字)	"	2	B	R	b	r
0x3	(制御文字)	(制御文字)	#	3	C	S	c	s
0x4	(制御文字)	(制御文字)	$	4	D	T	d	t
0x5	(制御文字)	(制御文字)	%	5	E	U	e	u
0x6	(制御文字)	(制御文字)	&	6	F	V	f	v
0x7	(制御文字)	(制御文字)	'	7	G	W	g	w
0x8	(制御文字)	(制御文字)	(	8	H	X	h	x
0x9	¥t (HT)	(制御文字)	)	9	I	Y	i	y
0xa	¥n (LF)	(制御文字)	*	:	J	Z	j	z
0xb	(制御文字)	ESC	+	;	K	[	k	{
0xc	(制御文字)	(制御文字)	,	<	L	＼	l	|
0xd	CR	(制御文字)	-	=	M	]	m	}
0xe	(制御文字)	(制御文字)	.	>	N	^	n	~
0xf	(制御文字)	(制御文字)	/	?	O	_	o	(制御文字)

数値計算の誤差(1/2)

- floatは内部で有効桁数の2進数で計算しているため，誤差が生じる.
- 数値をどこかの桁で四捨五入や切り捨てなどを行うことで誤差が生じる.

gosa1.c

```
#include <stdio.h>

int main(void)
{
  float sum=0.0;
  int i;

  for (i=0; i<100; i++)
    sum += 0.1;
  printf("sum=%f¥n", sum);

  return 0;
}
```

数値計算の誤差(2/2)

- 絶対値の大きい数字と小さい数字の加減では絶対値の小さい数字が無視されてしまう場合がある.
- 演算の順番を考慮する必要がある.

gosa2.c

```
#include <stdio.h>

int main(void)
{
  float num1, num2, num3, num4;
  float sum=0.0;

  num1 =  100002.0;
  num2 =       1.234567;
  num3 =       6.654321;
  num4 = -100001.0;

  sum += num1;
  sum += num2;
  sum += num3;
  sum += num4;
  printf("%f¥n", sum);
  return 0;
}
```

非線形方程式の数値計算

- 解の公式が存在しないような方程式を数値計算で近似解を求める
  - 2分法
  - ニュートン法

  など

2分法

① 実関数 $f(x)$が区間 $[a,b]$ で連続

② $f(a)f(b) < 0$

③ $[a,b]$ に解は1つしかない（中間値の定理により，解は少なくとも1つは存在する）

このとき，$f(x)=0$の解αを求める．

2分法のアルゴリズム

1. $x_1^{(0)} = a,\ x_2^{(0)} = b$ とおく. $f(x_1^{(0)})f(x_2^{(0)}) < 0$ である.
2. 反復手順:

i=1,2 .....に対して

$$x^{(i)} = \frac{1}{2}(x_1^{(i-1)} + x_2^{(i-1)})$$

$f(x_1^{(i-1)})f(x^{(i)}) < 0$ ならば $x_1^{(i)} = x_1^{(i-1)},\ x_2^{(i)} = x^{(i)}$

$f(x_1^{(i-1)})f(x^{(i)}) > 0$ ならば $x_1^{(i)} = x^{(i)},\ x_2^{(i)} = x_2^{(i-1)}$

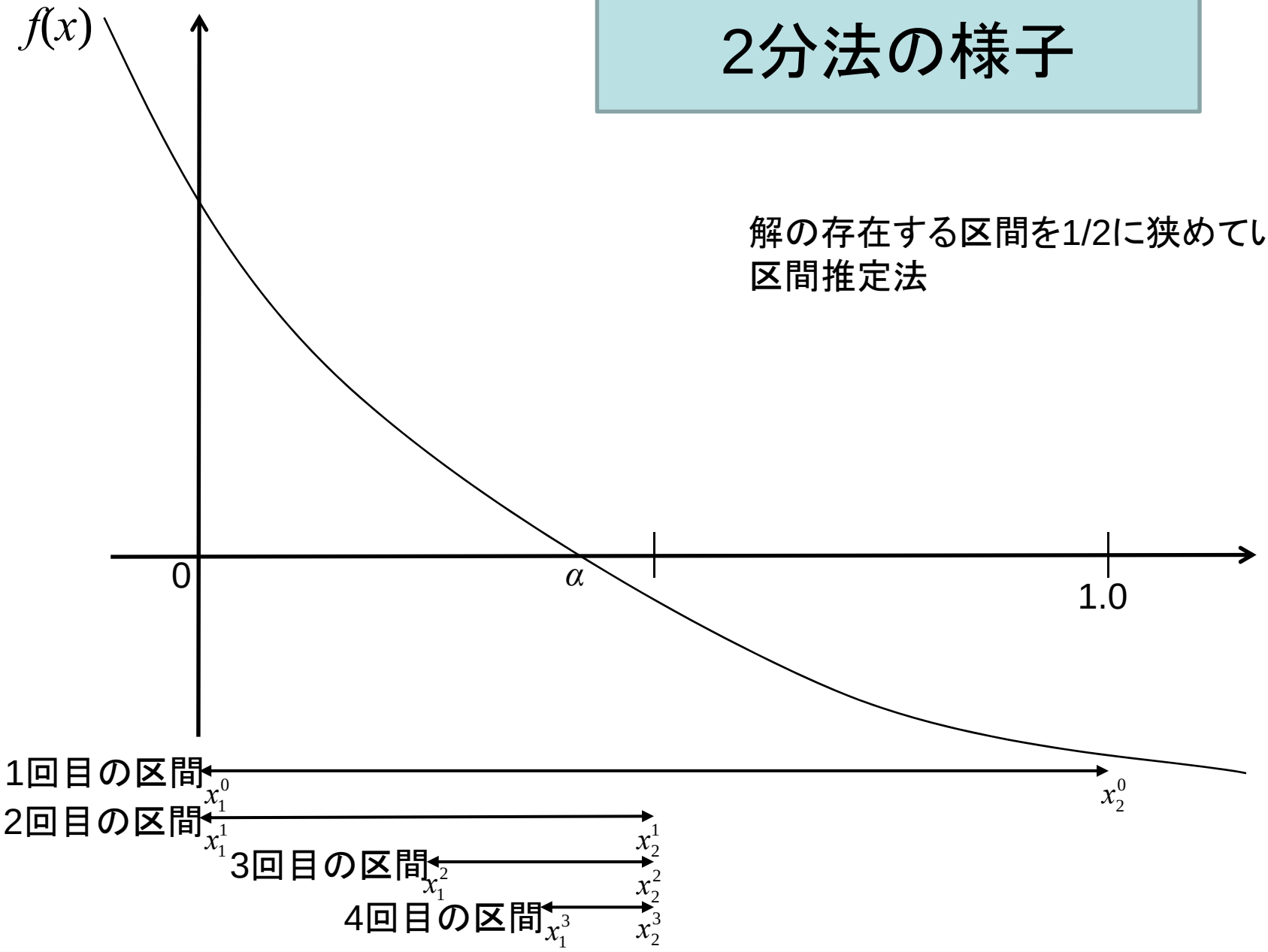
を繰り返す

3. 停止条件: 十分小さな値εについて

$\left|x_1^{(i)} - x_2^{(i)}\right| < \varepsilon$ のとき反復終了($(x_1^{(i)} + x_2^{(i)})/2$ が解).

2分法の様子

解の存在する区間を1/2に狭めていく
区間推定法



本日の演習(1/2)

- gosa1.cを入力し，実行結果をkadai5_1.txtとして提出せよ.
- gosa2.c において，誤差が出ないように演算の順番を修正せよ(kadai5_2.c).
- 次ページのプログラムを参照し，2分法のプログラムを作成し，$f(x)=x^3-3x+1=0$の$[0,1]$区間の解αを求めよ(kadai5_3.c). ただし，$\varepsilon=2^{-30}$ とする.

参考

kadai5_3.c

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

double bisec(double a, double b, double eps);
double f(double x);

int main(void)
{
  double a,b,x,y,eps;


  eps=pow(2.0, -30.0);
  a=0.0; b=1.0;

  printf("answer = %f¥n", bisec(a,b,eps));

  return 0;

}

/* 右へ続く */
```

kadai5_3.c

```
double bisec(double a, double b, double eps)
{
  ........
  /* 2分法の本体. ここを作成する */
  ........
}


double f(double x)
{
  return (......); /* f(x)の本体. ここも作成*/
}
```

本日の演習(2/2)

- 次頁kadai5_4.cをベースに，book構造体の配列itemについて
  - 最大のページ数を持つ本のタイトルを表示
  - 最小の価格の本の情報を表示

  するようにプログラムを完成させよ.
- 文字列を引数にとり，その文字列の長さを返す関数str_len，文字列2つを引数にとり，2つ目の文字列を1つ目にコピーする関数str_cpyを作成せよ．ただし，strlenやstrcpyは使わないこと(kadai5_5.c).

※ Course N@viにてプログラムを提出

(締め切り：本日中)

参考

kadai5_4.c

```
#include <stdio.h>
#include <string.h>

typedef struct {
  char title[30];
  int pages;
  int price;
} BOOK;

BOOK item[8]={
  {"English Reading", 110, 980},{"C Prog.", 218, 3000},
  {"Perl Prog.", 120, 1800}, {"C++ Prog.", 360, 4500},
  {"Pascal Prog.", 190, 2200},{"Lisp Prog.", 150, 1700},
  {"JAVA Prog.", 200, 2900},{"C# Prog.", 240, 3800}};
int item_num=8;

int main(void)
{
  int i=0;
  int max_pages, max_pages_id;
  int min_price, min_price_id;
```

kadai5_4.c

```
  /*
     最大のページ数を持つ本を探す
     価格が最も安い本を探す
  */


  printf("max pages book: %s¥n",
          item[max_pages_id].title);
  printf("min price book: %s, pages:%d,
                  price: %d¥n",
     item[min_price_id].title,
     item[min_price_id].pages,
     item[min_price_id].price);

  return 0:
}
```

次回予定

- 非線形方程式の解法(2)
  - ニュートン法