シミュレーション基礎(5)

第3章　流れを管理する

3.1　forで繰り返す

3.1.1 和を求める

(1) $s_1 = 1 + 2 + 3 + \cdots + n$

(2) $s_2 = 1^2 + 2^2 + \cdots + n^2$

```
S1=0; S2 = 0;
for I=1:100
  S1=S1 +I;    S2=S2+I.^2;
end
S1，S2
```

または

```
   S1=0; S2 =0; Loop=1:100;
   for I=Loop
     S1=S1+I; S2=S2+I.^2;
   end
 S1，S2
```

3.1　forで繰り返す

3.1.2 for文をネストする

For文の中にfor文を書くことができる
(例) 行列の成分の和を求める

```
 A=[1  2   3   4;   5   6  7   8;   9    1 0   11   12];
S=0;
for I=1:3
      for J=1:4
        S=S+A(I,J);
      end
end
  S
```

3.2　whileで繰り返す

3.2.1 和を求める

```
S1=0; S2 =0; i= 0;
while   i<100
  i=i+1;   S1 = S1 +i ;  S2=S2+i.^2;
end
S1,   S2
```

```
 while 終了条件
          実行文たち
   end
```

- 終了条件には真偽が問える命題が置かれる.
- 命題は，関係演算子＋論理演算子で作られる.

関係演算子：

x==y　：ｘはｙと等しい
x~=y　:ｘはｙと等しくない
x＞y　　:ｘはｙより大きい
x＞=y　:ｘはy以上
x＜y　　:ｘはｙより小さい
x＜=y　:ｘはy以下

- 論理演算子：

~p　　：ｐの否定
p&q　　:ｐとｑの論理積(かつ)
p|q　　：ｐとｑの論理和（または）
ｐ xor ｑ　：ｐとｑの排他的論理和
（ｐのみまたはｑのみ真）

命題の真偽はそれぞれ１または0で表される．なお，零でない数値は真であるとみなされる.

3.2　whileで繰り返す

3.2.2 while文をネストする

```
A=[1 2 3 4; 5 6 7 8; 9  10  11  12]；
 S=0;i=0;
while i<3
i=i+1;
j=0;    while j<4;    j=j+1;    S= S + A(i,j) ;    end
end
  S
```

3.3　どちらが速いか

3.3.1 計測時間を測る

tic：タイマーをスタート
toc：タイマー停止

同じ計算をする３つのプログラムの計算時間を比べてみよう

```
clear all;   N=100000；
  % Program  1 ：for
  tic
 S1=0;   S2 =0;
    for   I=1：N;
   S1=S1+I;
   S2=S2+I.^2;
 end
  T1=toc
```

```
  % Program 2:while
  tic
  S1=0;   S2=0;   I= 0;
while I<N;   I=I+1;
   S1=S1+I;
   S2=S2+I.^2;
   end
 T2=toc
 % Program 3 : sum
    tic
      X=1:100000;
S1=sum(X) ;   S2=sum(X.^2) ;
T3=toc
```

組込み関数を用いると計算が早い

3.3　どちらが速いか

3.3.2 領域の確保と並列化

```
clear all;
N=10001;
% Program 1 : for
 tic
for i=1:N+1;Y(i)=sin((i-1)./N);end
 T1=toc
 % Program 2:  zeros
 tic
 Y=zeros(1,N+1);
for i=1:N+1;Y(i)=sin((i-1)./N) ;end
T2=toc
% Program 3 ： sum
tic
X=0:1./N:1; Y=sin(X);
 T3=toc
```

メモリに保持している変数をクリアする
（別なプログラムを走らせるとき行った方がよい）

すこしずつ記憶領域を拡張

必要な記憶領域を最初に確保

必要な記憶領域を最初に確保

必要なメモリを最初に確保すると早い

3.3　どちらが速いか

3.3.3 インデックス利用による高速化

同じ計算をする３つのプログラムの
計算時間を比べてみよう

```
EX30303:
  clear all;    N=100000；
  % Program  1 ：for
  tic
S1=0;    S2 =0;
    for    I=1：N;
   S1=S1+I;
    S2=S2+I.^2;
end
 T1=toc
```

3.4　ifで分岐する

3.4.1 実行したりしなかったり

基本の構文

```
If  判定条件
        実行文たち
end
```

```
X=3;
 if X>0 ; disp( 'X is positive. ' ) ;  end
X=-4;
 if X>0; disp( 'X is positive.'); end
```

dispはコマンドウィンドウに変数名などを省いて行列や文字列を表示するための組み込み関数

3.4　ifで分岐する

3.4.2 こっちやあっちを実行したり

基本の構文　＃２

```
If　判定条件
        実行文たち1
else
        実行文たち２
end
```

```
X= -2;
 if  X>0
disp( 'X is positive.');
 else
disp( 'X is not positive.');
  end
```

3.4　ifで分岐する

3.4.3 3つに分けて実行する

```
If　判定条件１
        実行文たちI
 elseif　判定条件２
        実行文たち2
 else
        実行文たち３
 end
```

```
  X=0;
 if X>0
disp('X is positive.');
  elseif X<0
disp('X is negative.');
 else
  disp('X is zero.');
 end
```

3.5　switchで分岐する

3.5.1 値で場合分けをする

```
  X=0;
switch sign(X)
case 1
disp( 'X is positive.')
case  -1
disp( 'X is negative.');
otherwise
disp('X is zero.');
end
```

Case には複数の値をおくこともできる

```
X=-3;
  switch sign(X)
  case  {1,-1}
        disp (   'X is not zero.');
  otherwise
        disp( 'X is   zero.');
 end
```

3.6　ループの中止と継続

3.6.1 みつかれば中止する

```
T0=clock;
  while etime(clock,T0)<5
   X=fix(rand.*10000) ;
   if X==2000
   X,  break
   end
end
```

clockは現在の時刻を年，月，日，時，分，秒を成分とするベクトル

etimeはclockで与えられる2つの時刻の差を秒で与える

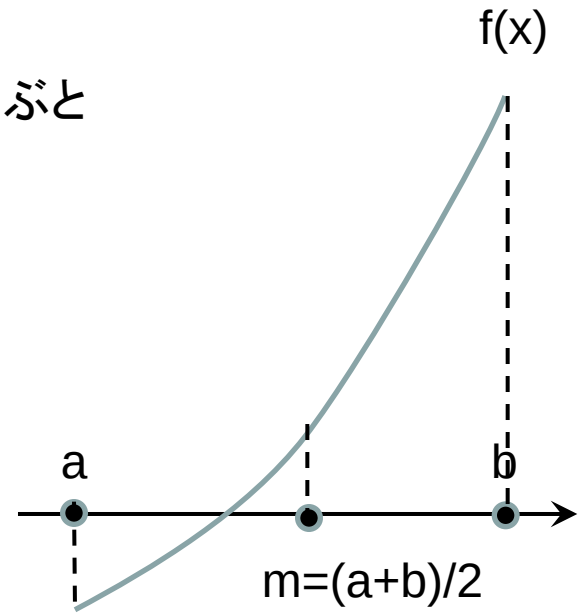
randは0以上１未満の一様乱数を発生する関数

fixは数を切り捨てて整数にする関数

したがってfix(rand.*10000)は0以上9999以下の整数ができる

breakによってループの外側に抜け出る

応用(1)　　２分法



方程式 f(x)=0 の解を挟む適当な初期値a,b　(a<b) を選ぶと

$f(a)\cdot f(b)<0$

が成り立つ。a,bの中間点

$m=(a+b)/2$

を計算しf(m)を計算する。

$f(m)=0$

ならmが解であるから。一方、

$f(a)\cdot f(m)<0$

なら解はaとmの間にあるということだから

$b=m$

とする。正解を見つけるか、a,bの距離が十分十分小さくなるまで計算を続ける。

f(x)=x^3+2x-2のときf(x)=0の解を2分法で求めたい。以下の計算を実行しなさい。

1. $0 \le x \le 2$ に0.05間隔に等間隔で分布するベクトルxを作成しなさい。

2. 1.で作成したxの各点に対するf(x)を計算し、その結果をベクトルyとしなさい。

   y=x.^3+2*x-2

3. (x,y)をプロットして曲線の形を調べなさい。

4. 3.の結果より2分法のための初期値a,bを決定しなさい。

5. .2分法によりf(x)=0の解を求めなさい。

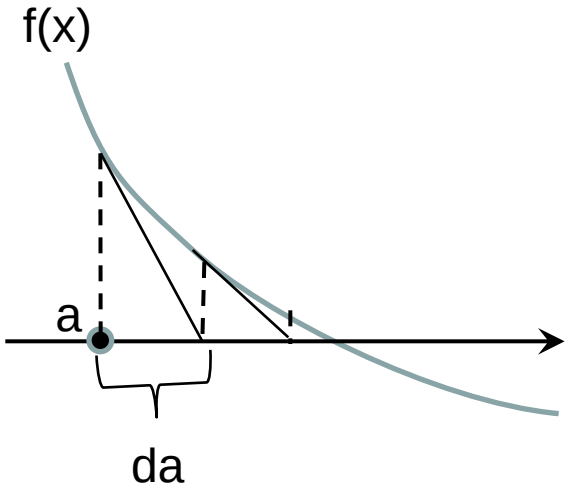
```
a=0;
b=2;
while abs(a-b)>1.0e-4
    m=(a+b)/2
    fm=m^3+2*m-2;
    fa=a^3+2*a-2;
    fb=b^3+2*b-2;
    if fm==0
      break;
    
      b=m;
    
      a=m;
    end
end
```

応用(２)　　Newton法

2分法より収束が早い計算法としてNewton法がある．そのアルゴリズムは以下のとおり．

1. f(x)=0の解として初期値x=aを選ぶ.
2. f(a)を計算する.
3. ｜f(a)｜が十分小さければaを解として計算を終了する.
4. (a, f(a))で接線を引き，接線とx軸が交わる点を求める.

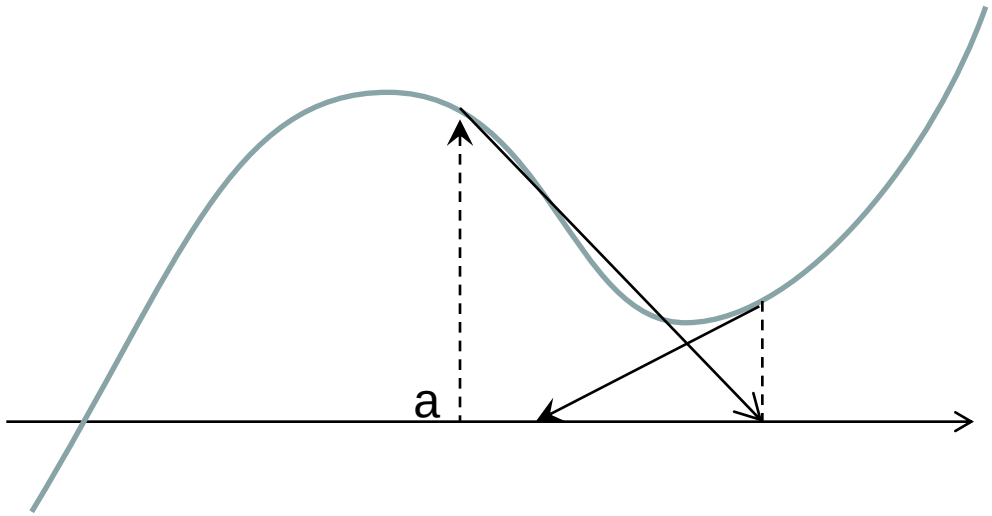
   da=－f(a)/f’(a)
5. 新たな近似解を a-> a+da とし2.へ



$f(x)=x^3+2x-2$のとき$f(x)=0$の解を$x=0$を初期値にNewton法で求めなさい.

```
a=0;
fa=1;
while abs(fa)>1.0e-4
    fa=                ;
    fd=
    da=-fa/fd ;
   a=
 end
```

Newton法は初期値が解の十分近傍にあることが必要



Newton法が収束しない例