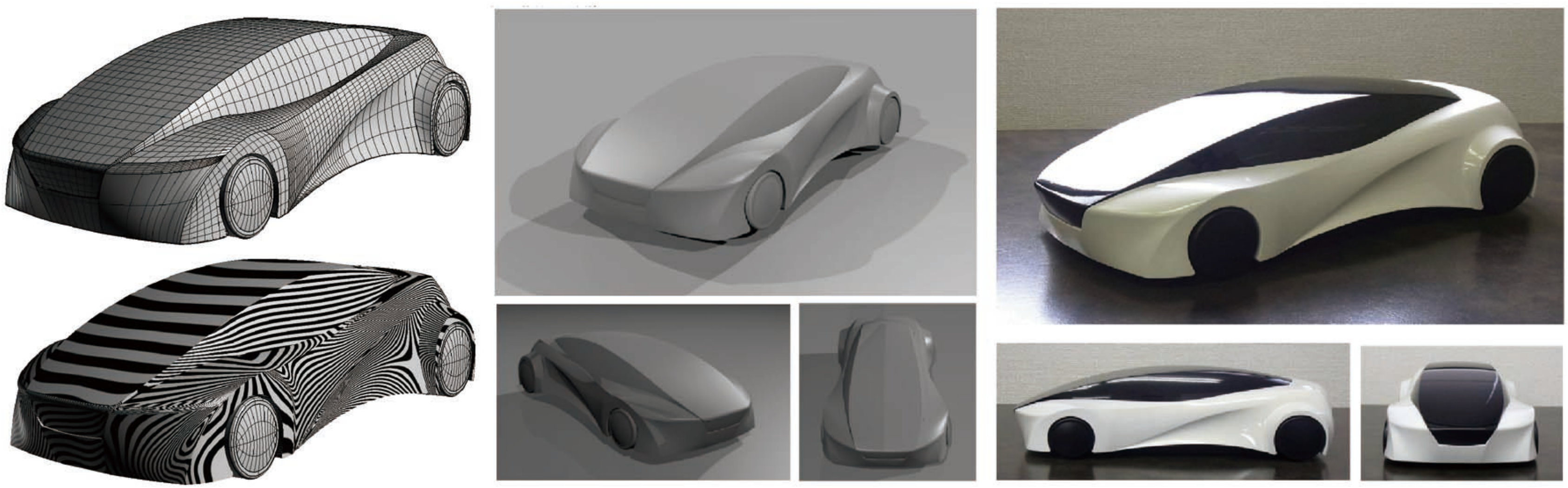
数値解析

2022年度前期　第5週 [5月19日]



(a) Isoparametric lines and zebra mapping　　(b) Rendering　　(c) Mock-up

三浦 憲二郎

講義アウトライン [5月19日]

- 非線形方程式
  - 復習
    - ２分法

- 非線形方程式
  - ニュートン法 （1変数）

復習：非線形方程式　p.70

•代数方程式の解　$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = 0$

•4次方程式までは公式があり，解ける.

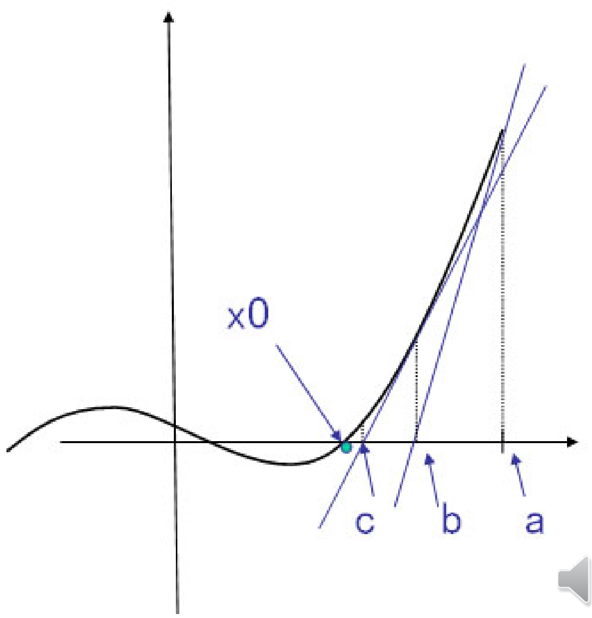
•5次以上の方程式に代数的解法は存在しない.

•数値的に解くしかない.

•高次の連立方程式，三角関数など変数のn乗で表せない項を含む方程式

$$f(x) = x - \cos x = 0$$

•数値的に解くしかない.



復習：2分法 p.70

- 定理4.1 （中間値の定理） 関数$f(x)$は閉区間$[a,b]$で連続で $f(a) \neq f(b)$ ならば，$f(a)$ と $f(b)$ の間の任意の数 k に対して $f(c)=k$ となる c $(a<c<b)$が存在する.

2分法

- 中間値の定理より，$f(a)\,f(b) < 0$ ならば，$f(c)=0$，$a < c < b$ となる c が存在する.
- (1) 何らかの方法で $f(a)\,f(b) < 0$ となる閉区間 $[a, b]$ を求める.
- (2) $c_1 = (a+b)/2$
  - (2a) $f(a)\,f(c_1) < 0$ のとき，解 α は $[a, c_1]$ に存在する.
  - (2b) $f(c_1)\,f(b) < 0$ のとき，解 α は $[c_1, b]$ に存在する.

  (2a) では $[a, b]$ の代わりに$[a,c_1]$, (2b)では $[c_1, b]$とする.
- ステップ(2)に戻り，閉区間が十分に小さくなるまで繰り返す

復習:2分法のアルゴリズム p.71

Input a, b, ϵ

Do

$c \leftarrow \frac{a+b}{2}$

if $f(a)f(c) < 0$ then

$b \leftarrow c$

else

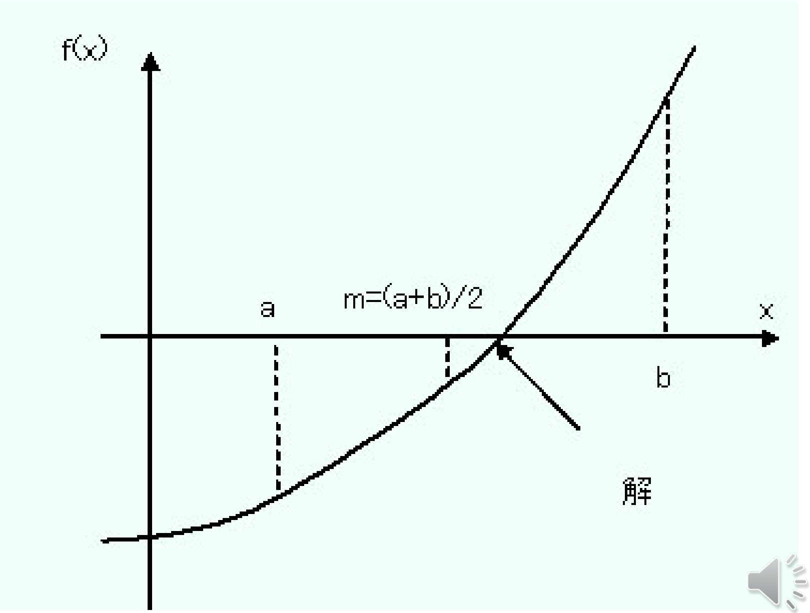
$a \leftarrow c$

end if

while($|a-b| \geq \epsilon$)

$c \leftarrow \frac{a+b}{2}$

Output c



復習:2分法

区間 [a, b]に解が1つしか存在しないのならば,

- n回目の区間 $$d_n = \frac{d_{n-1}}{2} = \frac{d_{n-2}}{2^2} = \cdots = \frac{d_0}{2^n}$$
- d_n < ε のときに計算を打ち切るとすると，必要な計算回数nは,

$$\begin{aligned} d_n &= \frac{d_0}{2^n} < \epsilon \\ n &> \frac{\log(\frac{d_0}{\epsilon})}{\log 2} \end{aligned}$$

を満たす最小の自然数

絶対値誤差 $$|c_n - \alpha| < \frac{d_n}{2}$$

ステップ1) 区間 [a, b] の求め方

- (1) 最初の区間 [x_{min}, x_{max}]，および 微小区間の幅 h を与える.
- (2) $n=(x_{max}-x_{min})$ / h として分割数を定め，$x_0=x_{min}$とする.
- (3) k=1,2,･･･,nに対して，$x_k=x_{min}+kh$とし，$f(x_{k-1})\,f(x_k) < 0$ なら [x_{k-1}, x_k] を対象区間にする.

復習:2分法:プログラム

```
#include <stdio.h>
#include <math.h>
double bisection(double a,double b,double eps); /* 2分法 */
double f(double x); /* 関数の定義 */
int main(void){
    double a,b,x,h,y1,y2,eps=pow(2.0,-30.0);
    int n;

    printf("初期区間[a,b]を入力してください. ---> a b¥n");
    scanf("%lf%lf",&a,&b);
    printf("区間の分割数nを入力してください. ---> n¥n");
    scanf("%d",&n);

    /* 対象区間を探索しながら2分法を適用 */
    h=(b-a)/n; y1=f(a);
    for(x=a+h;x<=b;x+=h){
        y2=f(x);
        if(y1*y2<0.0){
            printf("求める答えはx=%fです.¥n",bisection(x-h,x,eps));
        }
        y1=y2;
    }
    return 0;
}
```

2分法:プログラム

```
/* 2分法 */
double bisection(double a,double b,double eps){
    double c;

    do{
        c=0.5*(a+b);
        if(f(a)*f(c)<0){
            b=c;
        }
        else{
            a=c;
        }
    }while(fabs(b-a) >=eps);/* fabs()は絶対値を返す.「C言語入門」p.264 */
    c=0.5*(a+b);
    return c;
}

/* 関数の定義 */
double f(double x){
    return x*(x*x*(x*x-5.0)+4.0); /* x*x*x*x*x-5.0*x*x*x+4.0*x */
}
```

復習：プログラム：実行結果

初期区間[a,b]を入力してください．－－－>a b

−3　3

区間の分割数nを入力してください．－－－> n

10

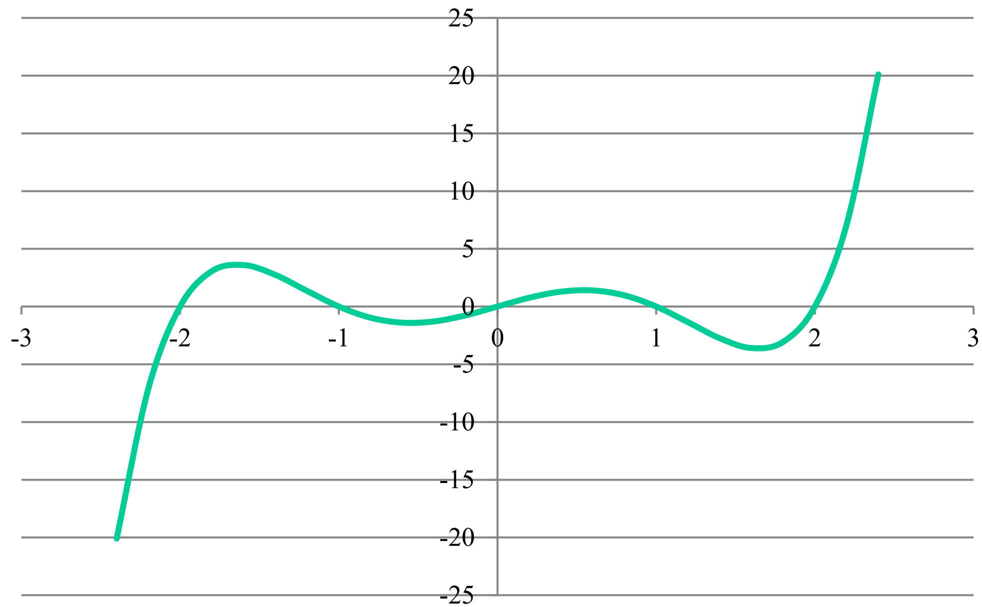
求める答えはx=−2.000000です．

求める答えはx=−1.000000です．

求める答えはx=−0.000000です．

求める答えはx=1.000000です．

求める答えはx=2.000000です．



反復法と縮小写像の原理　p.74

2分法　解の存在範囲を探索しながら狭めていく.　**直接探索法**

ニュートン法　初期値x_0から出発して解αに収束するような列$\{x_n\}$を作り，x_nがαに十分近づいたときに計算を打ち切る.

反復法，あるいは**逐次反復法**

- 反復法による列$\{x_n\}$

$$
\begin{aligned}
x &= g(x) && (1) \\
x_{n+1} &= g(x_n) && (2)
\end{aligned}
$$

g(x)：反復関数，式(1) x：不動点，式(2)：不動点反復

$\phi(x) \neq 0$となるように選んで，

$$
g(x) = x - \phi(x) f(x)
$$

$$
x = g(x) \;\Leftrightarrow\; \phi(x) f(x) = 0 \Leftrightarrow f(x) = 0
$$

ニュートン法 p.77

ニュートン法の原理

- 解 α の近傍で C^2級とする. (2次導関数が連続)
- テイラー展開
- ニュートン反復列

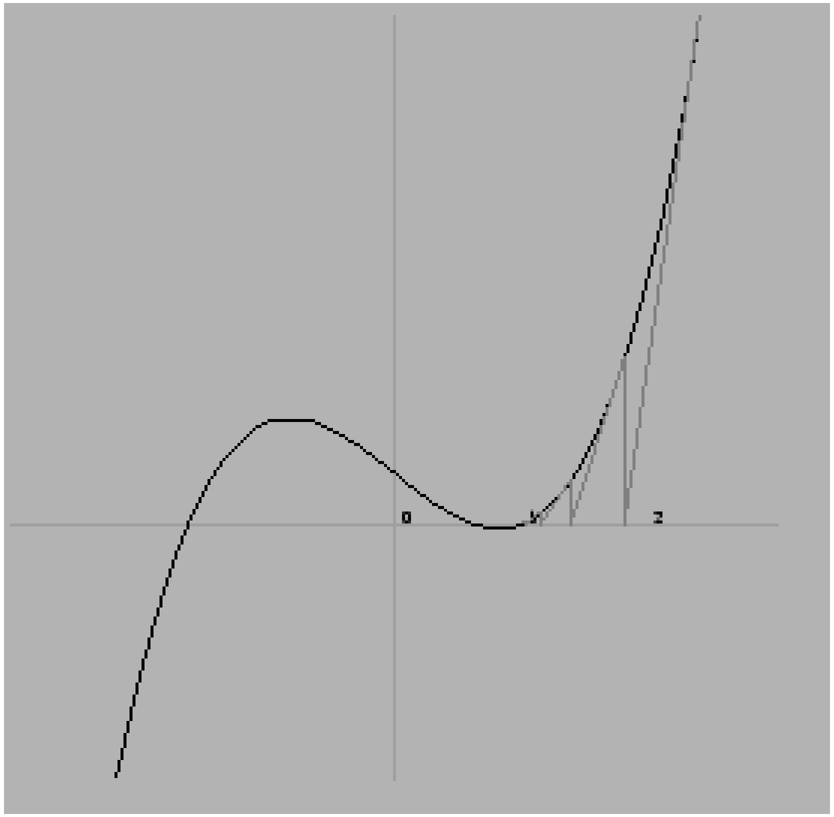
$$0 = f(\alpha) = f(x_0) + f'(x_0)(\alpha - x_0) + \frac{1}{2} f''(\xi)(\alpha - x_0)^2, \xi \in (x_0, \alpha) \text{ or } (\alpha, x_0)$$

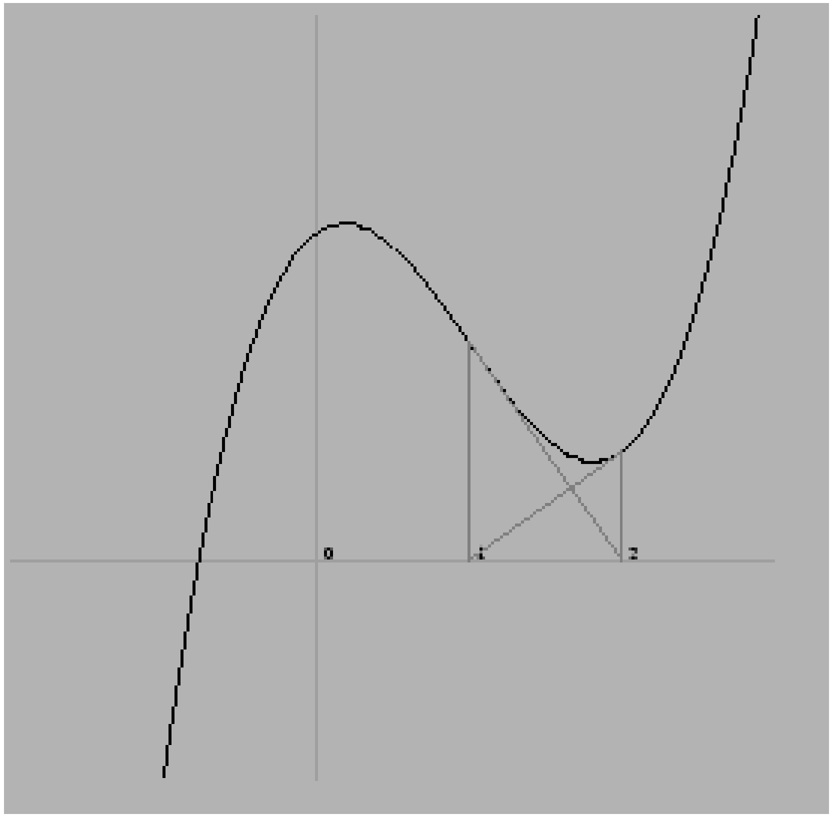
$$\begin{aligned} 0 &= f(\alpha) \approx f(x_0) + f'(x_0)\Delta x \\ f(x_0) + f'(x_0)\Delta x &= 0 \quad \text{Hence} \quad \Delta x = -\frac{f(x_0)}{f'(x_0)} \end{aligned}$$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

ニュートン法　p.77

収束しない場合がある.





ニュートン法:プログラム

```
#include <stdio.h>
#include <math.h>
#define EPS pow(10.0,-8.0) /* epsilonの設定 */
#define NMAX 10 /* 最大反復回数 */
void newton(double x); /* Newton法    */
double f(double x);     /* f(x)の計算  */
double df(double x);    /* f'(x)の計算 */
int main(void){
    double x;
    printf("初期値x0を入力してください¥n");
    scanf("%lf",&x);

    newton(x);
    return 0;
}
void newton(double x){/* Newton法 */
    int n=0; double d;

    do{
        d=-f(x)/df(x);
        x=x+d;
        n++;
    }while(fabs(d)>EPS&&n<NMAX);

    if(n==NMAX){
        printf("答えは見つかりませんでした¥n");
    }
    else{
        printf("答えはx=%fです. ¥n",x);
    }
}
/* 関数の定義 */
double f(double x){
    return x-cos(x);
}

double df(double x){
    return 1.0+sin(x);
}
```

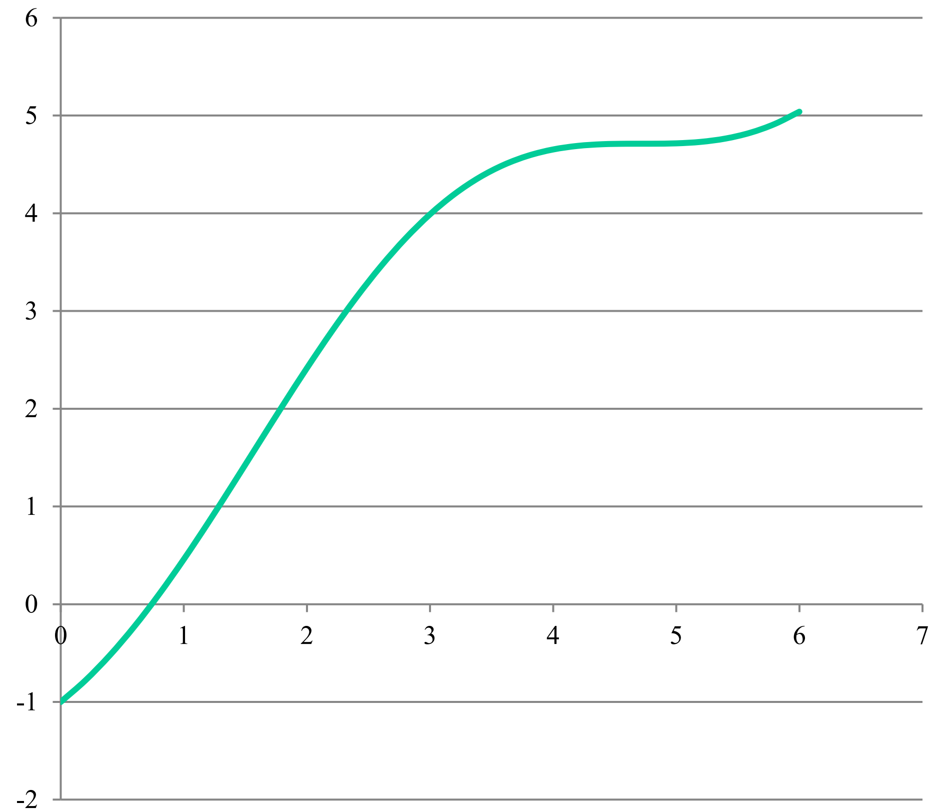
ニュートン法：実行結果　p.79

実行結果　　x−cos(x)=0

（1回目）
初期値x0を入力してください.
3
答えはx=0.739085です.

（2回目）
初期値x0を入力してください.
4
答えがみつかりませんでした.

（3回目）
初期値x0を入力してください.
5
答えがみつかりませんでした.



収束の速さ p.82

（1）線形収束

α に収束する反復列$\{x_n\}$が

$$(x_{n+1}-\alpha)=(A+\epsilon_n)(x_n-\alpha),\ |A|<1,\ \lim_{n->\infty}\epsilon_n=0$$

Aはnに依存しない定数

（2）p次収束

$$|x_{n+1}-\alpha|\le M|x_n-\alpha|^p,\ p>1,\ 0<M<\infty$$

Mはnに依存しない定数

収束の速さ p.82

定理4.3 f(x)=0の解x＝αが単解のとき，ニュートン法は2次収束する.

（証明）

f(α)＝0, f'(α)≠0 に注意すると，テイラーの公式より

$$
\begin{aligned}
x_{n+1} - \alpha &= x_n - \alpha - \frac{f(x_n)}{f'(x_n)} \\
&= \frac{-(f(x_n) + (\alpha - x_n)f'(x_n))}{f'(x_n)} \\
&= \frac{f''(\xi)}{2f'(x_n)}(\alpha - x_n)^2
\end{aligned}
$$

なぜならば，

$$f(\alpha) = f(x_n) + (\alpha - x_n)f'(x_n) + \frac{f''(\xi)}{2}(\alpha - x_n)^2$$

収束の速さ p.82

ニュートン法が収束するような区間において,

$$0 < A \leq |f'(x)|,\ |f''(x)| \leq B$$

となるような定数A, Bを選べば,

$$|x_{n+1} - \alpha| \leq \frac{B}{2A}|x_n - \alpha|^2$$

が成り立つ．これはニュートン法が（収束するならば）2次収束することを示している.

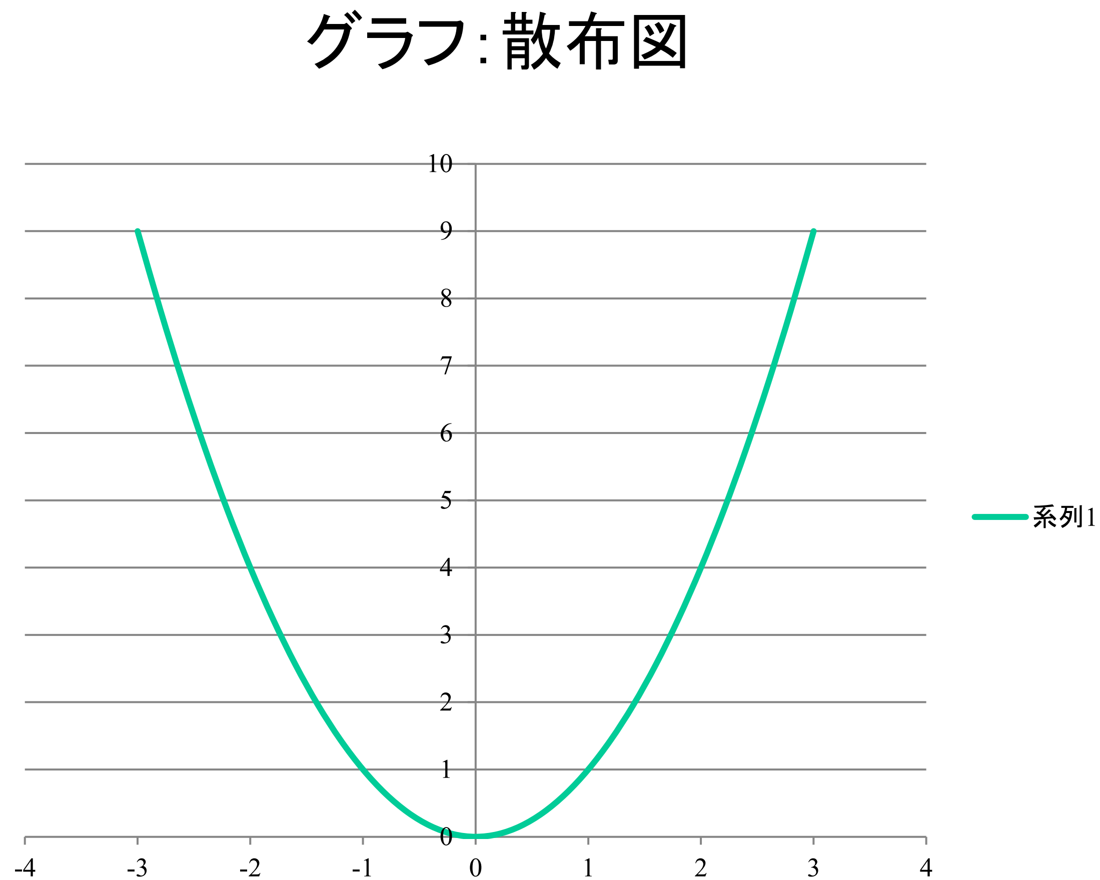
Excelによるグラフの作成

2次関数のグラフ

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int i; double x,y;
    FILE *fout;
    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);
    }
    for(x=-3;x<=3;x+=0.2){
        y=x*x;
        fprintf(fout,"%lf %lf¥n",x,y);
    }
    fclose(fout);
}
```

Excelによるグラフの作成

```
output.dat
-3   9
-2.87.84
-2.66.76
-2.45.76
-2.24.84
-2   4
-1.83.24
-1.62.56
-1.41.96
-1.21.44
-1   1
-0.80.64
-0.60.36
-0.40.16
-0.20.04
0    0
…
```



まとめ

- **非線形方程式**
  - **ニュートン法**

- **C言語の基礎**
  - ファイル入出力
  - Excelによるグラフの作成