ニュートン法による
非線型方程式の解

ニュートン法

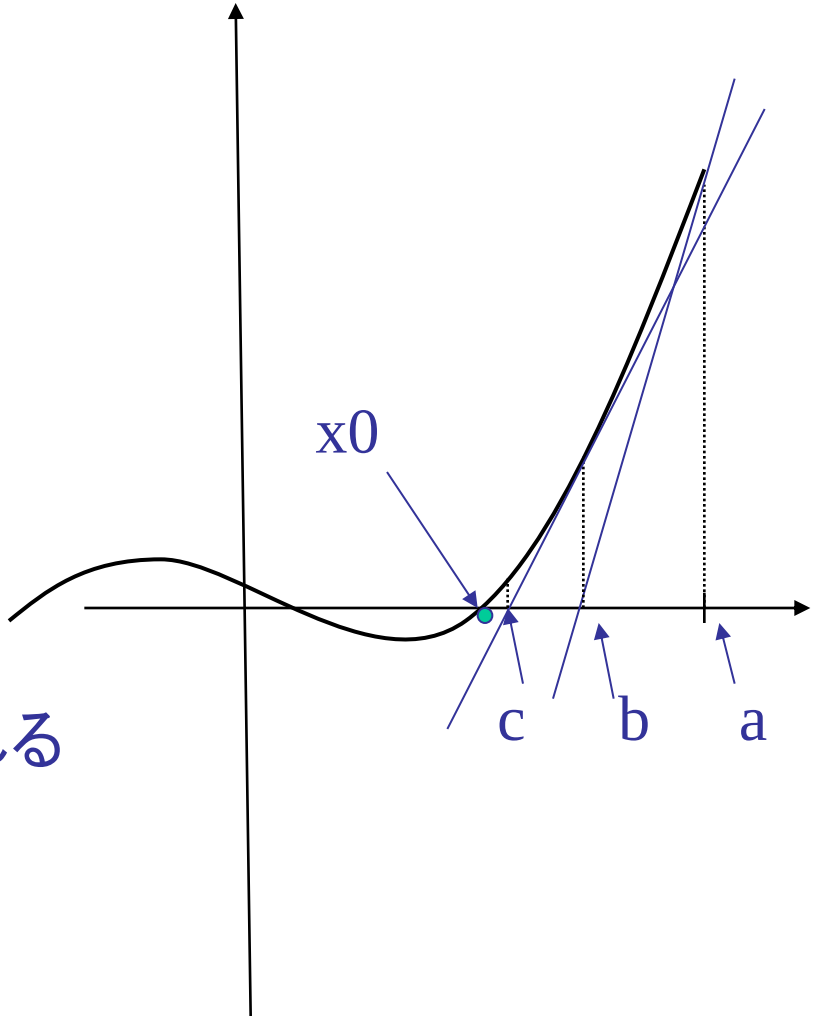
- 方程式f(x)=0の解を求める
- 処理手順
  1. 初期値 x の決定
  – $x = x_0$ を決める
  2. 接線と x 軸の交点の計算
  – $x = x_0$ における $y = f(x)$ の接線を引き、今度はこの接線と y=0 (x軸)の交点を x1 とする
  – すなわち、$x_{n+1} = x_n - (f(x)/f\,'(x))$ を計算する
  3. 2. を繰り返して値が収束したらそれを解とする

ニュートン法

初期値x=*a* をとる
グラフに接線1を引く
接線1が横軸をきる点x=b
*a*よりも真の値に近くなる

*b*のところで次の接線2を引く
Bよりも真の値に近い点 x=cが得られる

同じことを繰り返す
真の値x0に極めて近い値 を得る



ニュートン法の弱点

Newton 法は, 出発点とする十分近い解を見付けることができれば, 非常に収束が早い.

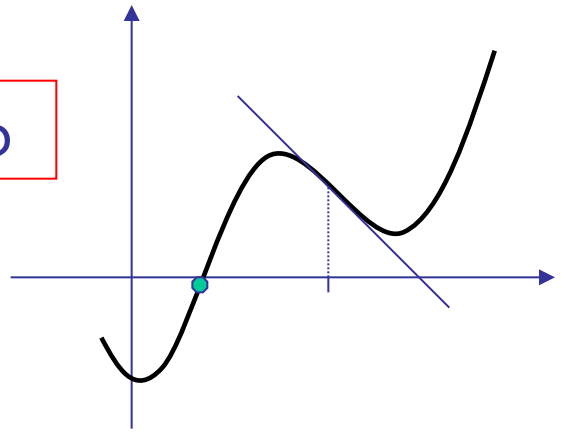
↕

初期値の選び方次第では収束しない

関数f(x)が単調でなくて変曲点を持つ,
つまりf’(x)の符号が変わるときには収束しない場合がある

三次以上になるとあまり有効な方法でなくなる.

無限ループが起こる



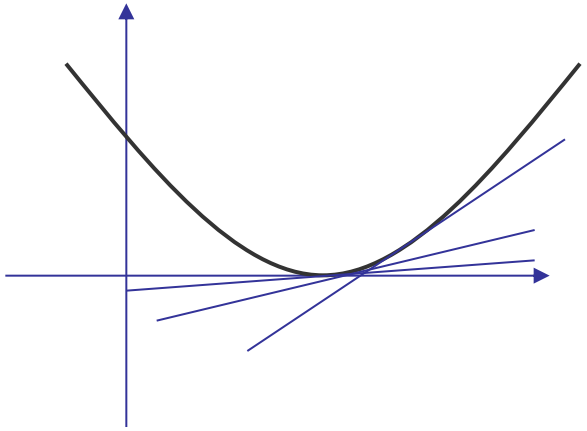
対策

繰り返しの上限回数を設定する

ニュートン法の弱点

重根の場合、誤差がなかなかゼロに収束しない

収束までに非常に時間がかかる



対策

収束条件を設定する

ニュートン法での収束条件

- ニュートン法では現在の x_n がどれだけ真の値に近いかは，一般には分からない
- 収束条件として、ある小さな正の数εに対して

$$\left| \frac{x_{n+1}-x_n}{x_{n+1}} \right| = \varepsilon$$

となった時点で計算を終了し x_{n+1} を解とする

ニュートン法のプログラム

- 入力
  - 初期値　x_0
  - 計算精度　ε
  - 方程式　f(x)
  - f(x)の導関数　f ’(x)
  - 繰り返し回数の上限　number
- 出力
  - 解(計算過程)

ニュートン法の注意点

- 初期値をいくつにするか?
  - 初期値の設定の際、あまりに解と掛け離れた値を与えると、収束するのに時間がかかったり、収束しなかったりする
- 収束条件をどうするか?
  - どの程度の精度で計算するかを決定していないと，繰り返しをいつ終えるか決まらない
- 収束しない場合はどうするか?
- 虚数解は求まらない

例題　ニュートン法のプログラム

- 初期値，計算精度、繰り返し上限回数を読み込んで，$f(x)=x^2-2$ をニュートン法で解くプログラムを作成する
  - $f(x) = x^2 - 2$
  - $g(x) = f'(x) = 2x$
  - x : 現在のx
  - new_x : 次のx
  - $g(x)$ が 10^{-4}以下なら重解とする

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

void function(void);
double f(double x);
double g(double x);

/* f(x)の出力 */
void function()
{
    printf("f(x) = pow(x,2) - 2¥n");
}

/* f(x) */
double f(double x)
{
    return pow(x,2) - 2;
}

/* f(x)の導関数 */
double g(double x)
{
    return 2 * x;
}
```

```
int main(void)
{
    double x;
    double new_x;
    double eps;
    int number, i;
    char buf[100];
    function();         /* f(x)を表示 */
    printf("初期値 : ");
    fgets(buf, 100, stdin);
    sscanf(buf,"%lf", &x);
    printf("計算精度 : ");
    fgets(buf, 100, stdin);
    sscanf(buf,"%lf", &eps);
    printf("繰り返し上限回数 : ");
    fgets(buf,100,stdin);
    sscanf(buf,"%d", &number);
```

```
    printf("繰り返し回数¥tnew_x¥t¥tf(x)¥t¥tg(x)¥n");
    for(i = 0 ; i < number ; i++) {
        new_x = x - f(x) / g(x);
        if(fabs(new_x - x) < eps * fabs(new_x)) {
            printf("x = %lf¥n",new_x);
            break;
        }
        printf("%2d¥t¥t%lf¥t%lf¥t%lf¥n",i,new_x,f(x),g(x));
        if(fabs(g(x)) < 1.0e-4){
            printf("x = %lf(重解)¥n",new_x);
            break;
        }
        x = new_x;
    }
    if(i == number)
    printf("繰り返し上限¥n");
}
```

実行結果の例

f(x) = pow(x,2) - 2
初期値 : 10
計算精度 : 0.000001
繰り返し上限回数 : 10

繰り返し回数	new_x	f(x)	g(x)
0	5.100000	98.000000	20.000000
1	2.746078	24.010000	10.200000
2	1.737195	5.540947	5.492157
3	1.444238	1.017846	3.474390
4	1.414526	0.085824	2.888476
5	1.414214	0.000883	2.829051

x = 1.414214

f(x) = pow(x,2) - 2
初期値 : 100
計算精度 : 0.000001
繰り返し上限回数 : 10

繰り返し回数	new_x	f(x)	g(x)
0	50.010000	9998.000000	200.000000
1	25.024996	2499.000100	100.020000
2	12.552458	624.250425	50.049992
3	6.355895	155.564203	25.104916
4	3.335282	38.397397	12.711789
5	1.967466	9.124103	6.670563
6	1.492001	1.870921	3.934931
7	1.416241	0.226067	2.984002
8	1.414215	0.005740	2.832483
9	1.414214	0.000004	2.828430

繰り返し上限