第１章　代数方程式 (fsolve)

1.1　概要

代数方程式の解 f(x)=0 を数値的に求めることを考える．標準的な

二分法 (bisection method) とニュートン法 (Newton's method)

の考え方と例を説明し，

収束性 (convergency) と安定性 (stability)

について議論する．さらに収束判定条件について言及する．

二分法のアイデアは単純．中間値の定理より連続な関数では，関数の符号が変わる二つの変数の間には根が必ず存在する．したがって，この方法は収束性は決して高くはないが，確実．一方，Newton 法は関数の微分を用いて収束性を速めた方法である．しかし，不幸にして収束しない場合や微分に時間がかかる場合があり，初期値や使用対象には注意を要する．

1.2　Maple での解

Maple では代数方程式の解は，fsolve で求まる．

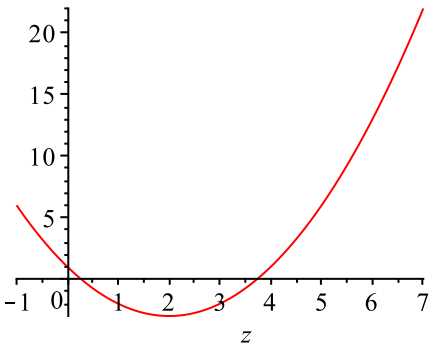
$$x^2 - 4x + 1 = 0$$

の解を考える．未知の問題では時として異常な振る舞いをする関数を相手にすることがあるので，先ずは関数の概形を見ることを常に心がけるべき．

```
> restart;
> func:=x->x^2-4*x+1;
```

$$func := x \mapsto x^2 - 4x + 1$$

```
> plot(func(z),z=-1..7);
```



もし，解析解が容易に求まるなら，その結果を使うほうがよい．Maple script の解析解を求める solve では，

```
> solve(func(x)=0,x);
```

と即座に求めてくれる．数値解は以下の通り求められる．

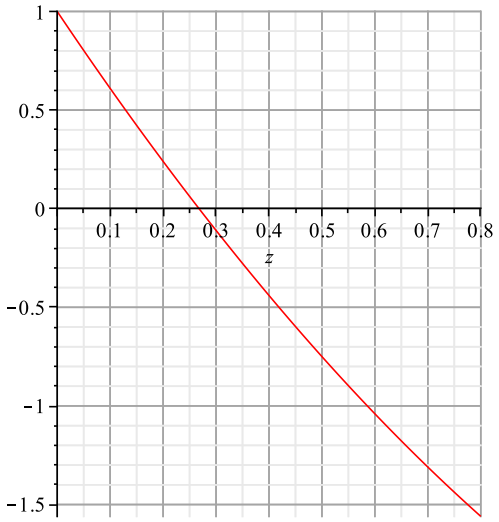
```
> fsolve(func(x)=0,x);
```


1.3 二分法と Newton 法の原理

1.3.1 二分法 (bisection)

二分法は領域の端 x_1, x_2 で関数値 $f(x_1), f(x_2)$ を求め，中間の値を次々に計算して，解を囲い込んでいく方法である．

```
> plot(func(z),z=0..0.8,gridlines=true);
```



x_1	x_2	$f(x_1)$	$f(x_2)$
0.0	0.8		

1.3.2 Newton 法 (あるいは Newton-Raphson 法)

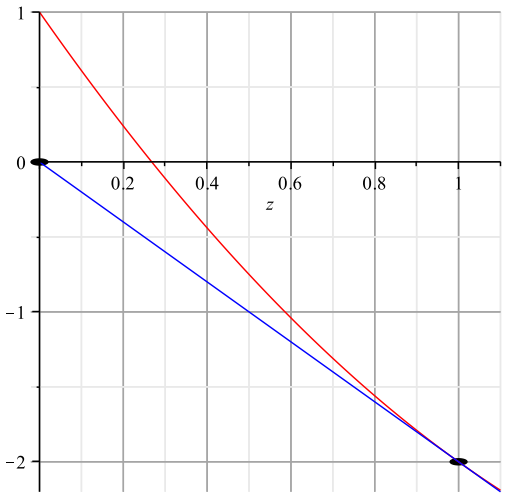
Newton 法は最初の点 x_1 から接線をひき，それが x 軸 (y=0) と交わった点を新たな点 x_2 とする．さらにそこでの接線を求めて...

という操作を繰り返しながら解を求める方法である．関数の微分を df(x) とすると，これらの間には

という関係が成り立つ．

```
> df:=unapply(diff(func(x),x),x);
```

```
> with(plots):with(plottools):
> x1:=1.0:x0:=0.0:
> p:=plot(func(z),z=0..1.1):
> p1:=plot(df(x1)*(z-x1)+func(x1),z=0..1.1,color=blue):
> p2:=[disk([x1,func(x1)],0.02), disk([x0,0],0.02)]:
> display(p,p1,p2,gridlines=true);
```



x_1	$f(x_1)$	$df(x_1)$
1.0		

1.4 二分法と Newton 法のコード

1.4.1 二分法 (bisection)

```
> x1:=0: x2:=0.8:
  f1:=func(x1): f2:=func(x2):
  for i from 1 to 5 do
    x:=(x1+x2)/2;
    f:=func(x);
    if f*f1>=0.0 then
      x1:=x; f1:=f;
    else
      x2:=x; f2:=f;
    end if;
    printf("%20.15f, %20.15f\n",x,f);
  end do:
```

```
0.400000000000000, -0.440000000000000
0.200000000000000,  0.240000000000000
0.300000000000000, -0.110000000000000
0.250000000000000,  0.062500000000000
0.275000000000000, -0.024375000000000
```

1.4.2 Newton 法 (あるいは Newton-Raphson 法)

```
> dfunc:=unapply(diff(func(z),z),z);
```

$$dfunc := z \mapsto 2z - 4$$

```
> x:=1: f:=func(x):
  printf("%15.10f, %+24.25f\n",x,f);
  for i from 1 to 5 do
    x:=x-f/dfunc(x);
    f:=func(x);
    printf("%15.10f, %+24.25f\n",x,f);
  end do:
```

```
1.0000000000, -2.0000000000000000000000000
0.0000000000, +1.0000000000000000000000000
0.2500000000, +0.0625000000000000000000000
0.2678571429, +0.0003188775510000000000000
0.2679491900, +0.0000000084726737970000000
0.2679491924, +0.0000000000000000059821834
```

以下のように Digits を変更すれば，Maple では浮動小数点演算の有効数字を変えることができる．

```
> Digits:=40;
```

$$40$$

1.5 収束性と安定性

実際のコードの出力からも分かる通り，解の収束の速さは 2 つの手法で極端に違う．2 分法では一回の操作で解の区間が半分になる．このように繰り返しごとに誤差幅が前回の誤差幅の定数 (< 1) 倍になる方法は 1 次収束 (linear convergence) するという．Newton 法では関数・初期値が素直な場合 ($f'(x) <> 0$) に，収束が誤差の 2 乗に比例する 2 次収束を示す．以下はその導出を示した．

```
> restart; ff:=subs(xi-x[f]=ei,series(f(xi),xi=x[f],4));
```

$$ff := f(x_f) + D(f)(x_f)\, ei + \frac{1}{2} D^{(2)}(f)(x_f)\, ei^2 + \frac{1}{6} D^{(3)}(f)(x_f)\, ei^3 + O\left(ei^4\right)$$

```
> dff:=subs({0=x[f],x=ei},series(diff(f(x),x),x,3));
```

$$dff := D(f)(x_f) + D^{(2)}(f)(x_f)\,ei + \frac{1}{2}D^{(3)}(f)(x_f)\,ei^2 + O\left(ei^3\right)$$

```
> ei1:=ei-ff/dff;
```

$$ei1 := ei - \frac{f(x_f) + D(f)(x_f)\,ei + \frac{1}{2}D^{(2)}(f)(x_f)\,ei^2 + \frac{1}{6}D^{(3)}(f)(x_f)\,ei^3 + O\left(ei^4\right)}{D(f)(x_f) + D^{(2)}(f)(x_f)\,ei + \frac{1}{2}D^{(3)}(f)(x_f)\,ei^2 + O\left(ei^3\right)}$$

```
> ei2:=simplify(convert(ei1,polynom));
```

$$ei2 := \frac{1}{3}\,\frac{3D^{(2)}(f)(x_f)\,ei^2 + 2D^{(3)}(f)(x_f)\,ei^3 - 6f(x_f)}{2D(f)(x_f) + 2D^{(2)}(f)(x_f)\,ei + D^{(3)}(f)(x_f)\,ei^2}$$

```
> ei3:=series(ei2,ei,3);
```

$$ei3 := -\frac{f(x_f)}{D(f)(x_f)} + \frac{f(x_f)\left(D^{(2)}\right)(f)(x_f)\,ei}{\left(D(f)(x_f)\right)^2} + \frac{1}{6}\,\frac{3\left(D^{(2)}\right)(f)(x_f) + 3\frac{f(x_f)\left(D^{(3)}\right)(f)(x_f)}{D(f)(x_f)} - 6\frac{f(x_f)\left(\left(D^{(2)}\right)(f)(x_f)\right)^2}{\left(D(f)(x_f)\right)^2}}{\left(D(f)(x_f)\right)}\,ei^2 + O\left(ei^3\right) \tag{1.1}$$

```
> subs(f(x[f])=0,ei3);
```

$$\frac{1}{2}\,\frac{D^{(2)}(f)(x_f)\,ei^2}{D(f)(x_f)} + O\left(ei^3\right)$$

注意すべきは，この収束性には一回の計算時間の差は入っていないことである．Newton 法で解析的に微分が求まらない場合，数値的に求めるという手法がとられるが，これにかかる計算時間はばかにできない．二分法を改良した割線法 (secant method) がより速い場合がある (NumRecipe9 章参照)．

二分法では，収束は遅いが，正負の関数値の間に連続関数では必ず解が存在するという意味で解が保証されている．しかし，Newton 法では，収束は速いが，必ずしも素直に解に収束するとは限らない．解を確実に囲い込む，あるいは解に近い値を初期値に選ぶ手法が種々考案されている．解が安定であるかどうかは，問題，解法，初期値に大きく依存する．収束性と安定性のコントロールが数値計算のツボとなる．

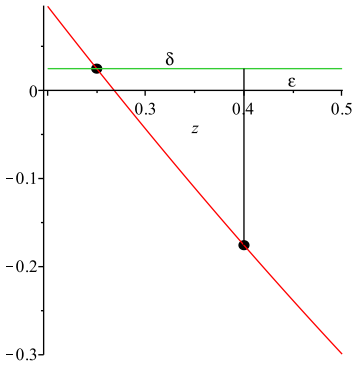
1.6 収束判定条件

どこまで値が解に近づけば計算を打ち切るかを決める条件を収束判定条件と呼ぶ．以下のような条件がある．

ε(イプシロン，epsilon) 法	
δ(デルタ，delta) 法	
占部法	数値計算の際の丸め誤差までも含めて判定する条件で， $\lvert f(x_{i+1})\rvert > \lvert f(x_i)\rvert$ とする．

ϵ, δ を説明するための図

```
> with(plots):with(plottools):
> f2:=x->0.4*(x^2-4*x+1):x1:=0.25:x0:=0.4:
  p1:=plot([f2(z),f2(x1)],z=0.2..0.5):
  p2:=[disk([x1,f2(x1)],0.005),disk([x0,f2(x0)],0.005)]:
  l1:=line([x0,f2(x0)],[x0,f2(x1)]):
  t1 := textplot([0.45,0.0,‘epsilon‘],align=above):
  t2 := textplot([0.325,0.05,‘delta‘],align=below):
> display(p,p1,p2,l1,t1,t2);
```



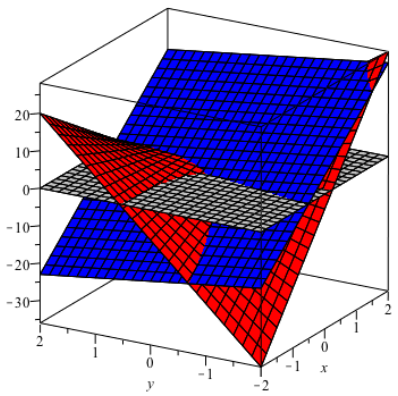
1.7　2 変数関数の場合

2 変数の関数では，解を求める一般的な手法は無い．この様子は実際に 2 変数の関数で構成される面の様子をみれば納得されよう．

```
> restart;
> f:=(x,y)->4*x+2*y-6*x*y; g:=(x,y)->10*x-2*y+1;
```

$$f := (x, y) \mapsto 4\,x + 2\,y - 6\,xy$$
$$g := (x, y) \mapsto 10\,x - 2\,y + 1$$

```
> p1:=plot3d({f(x,y)},x=-2..2,y=-2..2,color=red):
  p2:=plot3d({g(x,y)},x=-2..2,y=-2..2,color=blue):
  p3:=plot3d({0},x=-2..2,y=-2..2,color=gray):
  with(plots):
  display([p1,p2,p3],axes=boxed,orientation=[-150,70]);
```



解のある程度近くからは，Newton 法で効率良く求められる．

```
> fsolve({f(x,y)=0,g(x,y)=0},{x,y});
```

$$\{x = -0.07540291160, y = 0.1229854420\}$$

1.8 例題:二分法と Newton 法の収束性

代数方程式に関する次の課題に答えよ．(2004 年度期末試験)

1. $\exp(-x) = x^2$ を二分法およびニュートン法で解け．
2. n 回目の値 x_n と小数点以下 10 桁まで求めた値 $x_f = 0.7034674225$ との差 Δx_n の絶対値 (abs) の log を n の関数としてプロットし，その収束性を比較せよ．また，その傾きの違いを両解法の原理から説明せよ．

解答例

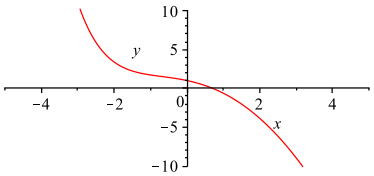
計算精度を 40 桁にしておく．func で関数を定義．

```
> restart; Digits:=40: func:=unapply(exp(-x)-x^2,x);
```

$$func := x \mapsto e^{-x} - x^2$$

先ずは，関数を plot して概形を確認．

```
> plot(func(x),x=-5..5,y=-10..10);
```



Maple の組み込みコマンドで正解を確認しておく．

```
> x0:=fsolve(func(x)=0,x);
```

$$0.70346742249839165204981860185990213034 29$$

テキストからプログラムをコピーして走らせてみる．環境によっては，printf 分の中の”\”が文字化けしているので修正．

```
> x1:=0: x2:=0.8: res1:=[]:
  f1:=func(x1): f2:=func(x2):
  for i from 1 to 20 do
    x:=(x1+x2)/2;
    f:=func(x);
    if f*f1>=0.0 then
      x1:=x; f1:=f;
    else
      x2:=x; f2:=f;
    end if;
    printf("%20.15f, %20.15f\n",x,f);
    res1:=[op(res1),[i,abs(x-x0)]]:
  end do:
```

```
0.400000000000000, 0.510320046035639
0.600000000000000, 0.188811636094026
0.700000000000000, 0.006585303791410
0.750000000000000, -0.090133447258985
0.725000000000000, -0.041300431044638
```

中略

```
0.703468322753906, -0.000001712107681
0.703467559814453, -0.000000261147873
```

プロットのためにリストを作成する．Maple でこれを実行するイディオムは以下の通り．

```
> res1:=[];
  for i from 1 to 3 do
    res1:=[op(res1),[i]];
  end do;
```

これを先のコードに組み込んでおいて得られた結果を logplot(片対数プロット) する．

```
> with(plots):
> logplot(res1); #res:結果は後に示す
```

同様に Newton 法での結果を res2 に入れる．

```
> dfunc:=unapply(diff(func(z),z),z);
```

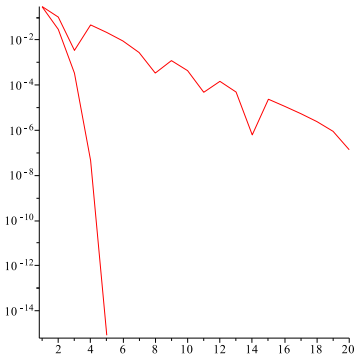
$$dfunc := z \mapsto -e^{-z} - 2\,z$$

```
> x:=1.0: f:=func(x):
  printf("%15.10f, %+24.25f\n",x,f);
  res2:=[[1,abs(x-x0)]]:
  for i from 2 to 5 do
    x:=x-f/dfunc(x);
    f:=func(x);
    printf("%15.10f, %+24.25f\n",x,f);
    res2:=[op(res2),[i,abs(x-x0)]]:
  end do:
```

```
1.0000000000, -0.6321205588285576784044762
0.7330436052, -0.0569084480040254074684576
0.7038077863, -0.0006473915387465014761973
0.7034674683, -0.0000000871660305624231097
0.7034674225, -0.0000000000000015809178420
```

res1, res2 を片対数プロットして同時に表示．

```
> l1:=logplot(res1);
> l2:=logplot(res2);
> display(l1,l2);
```

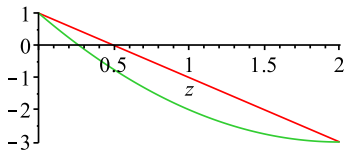


2 分法で求めた解は，Newton 法で求めた解よりもゆっくりと精密解へ収束している．これは，二分法が原理的に計算回数について一次収束なのに対して，Newton 法は 2 次収束であるためである．解の差 (δ) だけでなく，関数値 $f(x), \epsilon$ をとっても同様の振る舞いを示す．

1.9 課題

1. Newton 法の $f(x), df(x)$ の関係を示す式を導け．
2. 次の関数 $f(x) = \exp(-x) - 2\exp(-2x)$ の解を二分法，Newton 法で求めよ．
3. 代数方程式に関する次の課題に答えよ．(2004 年度期末試験)
   (a) $\exp(-x) = x^2$ を二分法およびニュートン法で解け．
   (b) n 回目の値 x_n と小数点以下 10 桁まで求めた値 $x_f = 0.7034674225$ との差 Δx_n の絶対値 (abs) の log を n の関数としてプロットし，その収束性を比較せよ．また，その傾きの違いを両解法の原理から説明せよ．
4. 次の方程式 $f(x) = x^4 - x - 0.12$ の正数解を二分法で求めよ．(2008 年度期末試験)
5. 収束条件がうまく機能しない例を示せ．
6. 割線法は，微分がうまく求まらないような場合に効率がよい，二分法を改良した方法である．二分法では新たな点を元の 2 点の中点に取っていた．そのかわりに下図に示すごとく，新たな点を元の 2 点を直線で内挿した点に取る．二分法のコードを少し換えて，割線法のコードを書け．また，収束の様子を二分法，Newton 法と比べよ．

```
> func:=x->x^2-4*x+1: x1:=0: x2:=2: f1:=func(x1): f2:=func(x2):
> plot({(z-x1)*(f1-f2)/(x1-x2)+f1,func(z)},z=0..2);
```



7. 次の方程式 $f(x) = \cos(x) - x^2$ の正数解を二分法で求めよ．割線法でも求め，収束性を比べよ．(2009 年度期末試験)
8. 次の方程式 $f(x) = x^3 - 3x + 3$ の解をニュートン法で求めよ．初期値をそれぞれ $x = -3, x = 2$ とした時を比べ，その差について論ぜよ．(2010 年度期末試験)