# Chap two

ShangXiaojin

2022 年 12 月 14 日

# 目录

# 1 Bisection

## 1.1 code

clc

clear

tic

format long

Bisection(1,2,-1,-2,1/2,1000,100,0.00001,0.00001)

toc

```
function [ans] = Bisection(Qc,Mc,C,a,b,N1,N2,Tol1,Tol2)
g = @(x)(Qc*x^2 + Mc*x + C);
i = 1;
m = a;
n = b;
k = 1;
while g(m)*g(n)>0
    m = rand*(b-a)+a;
    n = rand*(b-a)+a;
    i = i+1;
    if i>N1
        fprintf('you can increase N1 and try it again')
        break

    end
end
while k<= N2
    if abs(g((m+n)/2)) <= Tol1 || abs(m-n)<= Tol2
        ans = (m+n)/2;
        break
    else
        k = k+1;
        if g((m+n)/2)*g(m)>0
```

```
            m = (m+n)/2;

        else
            n = (m+n)/2;

        end
    end
end
k
end
```

### 1.2 ans

k = 18

ans = 0.414216995239258

历时 0.031189 秒

### 1.3 analyse

二分法需提前知晓 a，b 的值，使得 f(a)*f(b)<0, 且若范围内有多个零点概率丢失，做了一点改变使得可以随机找到满足条件的 a、b 两点。

## 2 Fixed-Point iteration

### 2.1 code

```
clear
tic
format long
g = @(x)(1-x^2)/2;
a = -2;
b = 1/2;
i = 1;
Tol = 0.00001;
```

```
N0 = 100;
p0 = (a+b)/2;
while i<=N0
    p = g(p0);
    if abs(p - p0)<Tol
        vpa(p)
        break
    else
        i = i+1;
        p0 = p;
    end
end
p0
i
toc
```

### 2.2 ans

ans = 0.41421521705482028385958415128698

i = 14

历时 0.066758 秒。

### 2.3 analyse

不动点法相对于二分法在该定义域和函数下步骤更简略一些，耗时也有所减少

## 3 Newton's

### 3.1 code

```
clear
tic
format long
```

```
syms x
y = x^2 + 2*x -1;
g = x - y/diff(y,x,1);
a = -2;
b = 1/2;
i = 1;
Tol = 0.00001;
N0 = 100;
p0 = (a+b)/2;
while i<=N0
    p = subs(g,x,p0);
    if abs(p - p0)<Tol
        vpa(p)
        break
    else
        i = i+1;
        p0 = p;
    end
end
i
toc
```

## 3.2 ans

ans = 0.41421356237309504884050067291276

i = 7

历时 0.313838 秒。

## 3.3 analyse

牛顿迭代仅用了七步，为得出结果所需步骤最少的不动点方法，但在MATLAB 中显然求导运算会耗费大量时间，时间上比其他方法有明显加长

# 4 Secant

## 4.1 code

```
clear
tic
format long
y = @(x)x^2 + 2*x -1;
a = -2;
b = 1/2;
i = 2 ;
p0 = a;
N0 = 100;
p1 = b;
Tol = 0.00001;
q0 = y(p0);
q1 = y(p1);
while i <= N0
    p = p1 - q1*(p1-p0)/(q1-q0);
    if abs(p-p1)<Tol
        p
        break
    else
        i = i+1;
        p0 = p1;
        q0 = q1;
        p1 = p;
        q1 = y(p);
    end
end
i
toc
```

### 4.2 ans

p = 0.414213562373142
i = 7
历时 0.062322 秒。

### 4.3 analyse

显然割线法与牛顿法都具有简介步骤的特点，但是相比较牛顿法的求导数，更新点的割线避开求导显然在耗时上有很大优势。

## 5 Method of False Position

### 5.1 code

```
clear
tic
format long
y = @(x)x^2 + 2*x -1;
a = -2;
b = 1/2;
i = 2 ;
p0 = a;
p1 = 2;
q0 = y(p0);
q1 = y(p1);
Tol = 0.00001;
N0 = 100;
while i<= N0
    p = p1 - q1*(p1-p0)/(q1-q0);
    if abs(p-p1)<Tol
        p
        break
    else
        i = i+1;
```

```
        q = y(p);
        if q*q1<0
            p0 = p1;
            q0 = q1;

        end
        p1 = p;
        q1 = q;
    end
end
i
toc
```

### 5.2 ans

p = 0.414210030816267

i = 16

历时 0.058421 秒。

### 5.3 analyse

与二分法有一定程度的类似，也是从闭合区间内接近根，不同的是通过两端点割线寻找根而不是二分区间。

## 6 Steffensen's

### 6.1 code

```
clear
tic
format long
g = @(x)(x - (x^2+2*x-1)/(2*x+2));
p0 = 1/2;
```

```
p1 = -2;
i = 1;
N0 = 100;
Tol = 0.000001;
while i <=N0
    p1 = g(p0);
    p2 = g(p1);
    p = p0 - (p1-p0)^2/(p2 - 2*p1 +p0);
    if abs(p - p0)<Tol
        p
        break
    else
        i = i+1;
        p0 = p;
    end
end
i
toc
```

## 6.2 ans

p = 0.414213562373095

i = 3

历时 0.035795 秒。

## 6.3 analyse

避开了计算导数但收敛极快。

# 7 Horner's

## 7.1 code

```
%Horner's f = x^4 - x^3 +x^2 -1
clear
tic
x0 = 1;
format long
A = [1 -1 1 0 1];
y = A(1);
z = A(1);
for i = 2:4
    y = x0*y + A(i);
    z = x0*z+y;0
end
    y = y*x0+A(5);
[y,z]
toc
```

### 7.2 ans

```
ans = 1×2
     2     3
历时 0.035833 秒。
```

### 7.3 analyse

Horner's 方法是减少乘法计算多项式值的方法，将其应用到先前的不动点等的运算中可以省时。