Computational Physics Assignment II

学号：515072910017

姓名：李进辉

Note：All code are in MATLAB R2017a

---

1. Consider the fixed-point iteration

Solution:

(a)

Suppose r is a fixed point of $g(x)$, then:

$|g'(r)| = \frac{1}{2}(1-\frac{a}{x^2})\Big|_{x=r} = \frac{1}{2}(1-\frac{a}{r^2})$

$g''(r) = \frac{a}{x^3}\Big|_{x=r} = \frac{a}{r^3}$

$r = \frac{1}{2}(r+\frac{a}{r}) \Rightarrow a = r^2$

In conclusion:

$\lim_{n\to\infty} \frac{|x_{n+1}-r|}{|x_n-r|} = \lim_{n\to\infty} |g'(\xi_n)| = |g'(r)| = 0$

$\lim_{n\to\infty} \frac{x_{n+1}-r}{(x_n-r)^2} = \frac{g''(r)}{2!} \neq 0$

So the convergence is quadratic, and for any initial point $x_1 > 0$, we have root $r = \sqrt{a}$, so the iteration converges to $\sqrt{a}$.

(b)

According to (a), $a = r^2$, so we simply let $a = 5$, then the root is $\sqrt{5}$, while $g(x) = \frac{1}{2}(x+\frac{5}{x})$, wirte code as follow:

```
function FixedPoint(f, x1)
% 初始化参数
tol = 1e-10; % 容许误差
N = 100;
% 开始循环
for n = 1:N
    x2 = f(x1);
    fprintf("N:%d \t x1:%.10f \t x2:%.10f\n", n, x1, x2);
    if abs(x2 - x1) < tol
        r = x2;
        fprintf("FixedPoint: The root %.10f was found after %d iterations.\n", r, n);

        return
```

```
    end
    x1 = x2;
end
```

CodeName:**FixedPoint.m**

Run the function:

```
>> f = @(x) 0.5 * (x + 5/x);
>> FixedPoint(f, 2.5)
N:1       x1:2.5000000000     x2:2.2500000000
N:2       x1:2.2500000000     x2:2.2361111111
N:3       x1:2.2361111111     x2:2.2360679779
N:4       x1:2.2360679779     x2:2.2360679775
N:5       x1:2.2360679775     x2:2.2360679775
FixedPoint: The root 2.2360679775 was found after 5 iterations.
```

And we can get the result is **2.2360679775**

(c)

We use function $f(x) = x^3 + x^2 - 5x - 5$, as we get the root of $f(x) = 0$, we can get the result of $\sqrt{5}$, write code as follow:

```
function Newton(f, x1)
syms x;
% 初始化参数
tol = 1e-10;         % 容许误差
diff_f = diff(f(x));  % f的导数形式
flag = 0;        % 用来判断导数为0的情况
N = 100;    % 默认循环次数
for n = 1:N
    x = x1;
    tmp = eval(diff_f);
    if tmp == 0      % 如果算到某一点的导数值为0，则返回失败信息
        break;
    end
    x2 = x1 - f(x1) / tmp;
    fprintf("N:%d \t x1:%.10f \t x2:%.10f\n", n, x1, x2);
    if abs(x2 - x1) < tol
        r = x2;
        flag = 1;
        break;
    end
    x1 = x2;
end

if flag == 1
    fprintf("Newton: The root %.10f was found after %d iterations.\n", r, n);
else

    fprintf("Convergence not found!\n");
```

```
end
```

CodeName:**Newton.m**

Run the function:

```
>> f = @(x) x^3 + x^2 - 5*x - 5;
>> Newton(f, 2.5)
N:1      x1:2.5000000000     x2:2.2666666667
N:2      x1:2.2666666667     x2:2.2365546635
N:3      x1:2.2365546635     x2:2.2360681036
N:4      x1:2.2360681036     x2:2.2360679775
N:5      x1:2.2360679775     x2:2.2360679775
Newton: The root 2.2360679775 was found after 5 iterations.
```

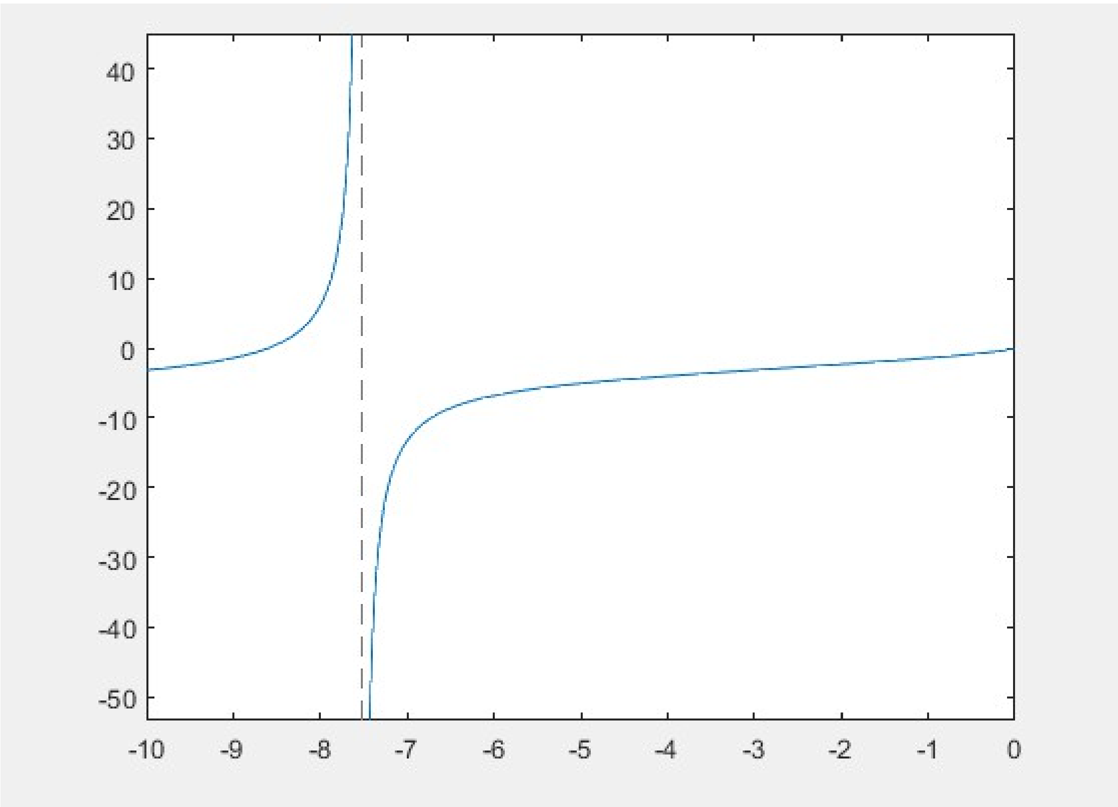
And we can get the result is **2.2360679775**

2. A particle of mass m is bound

Solution:

(a)

For even wave function:$\sqrt{10+E} * tan(\sqrt{10+E}) = \sqrt{-E}$, we define function $f(x) = \sqrt{10+x} * tan(\sqrt{10+x}) - \sqrt{-x}$, first use matlab to see the picture of this function:



We can see there is a root between -10 and -8, here we choose bisection method to calculate, code as follows:

```matlab
function bisection(f, a, b)
if f(a)*f(b) > 0
    fprintf("There is no root between %f and %f\n", a, b);
    return
end

% 初始化参数
tol = 1e-10;
N = 100;
% 开始循环
for n = 1:N
    c  = (a + b)/2;      % 取中点
    if f(c) == 0         % 如果为0，说明这个点就是要找的根
        break;
    end
    if (b-a)/2 < tol     % 如果小于所要求的精度就停止
        break;
    end
    if f(b)*f(c) > 0     % 判断符号是否改变
        b = c;
    else
        a = c;
    end
end
fprintf("Bisec: The root %.10f was found after %d iterations\n", c, n);
```

CodeName:**bisection.m**

After running this function, we can get the result as follow:

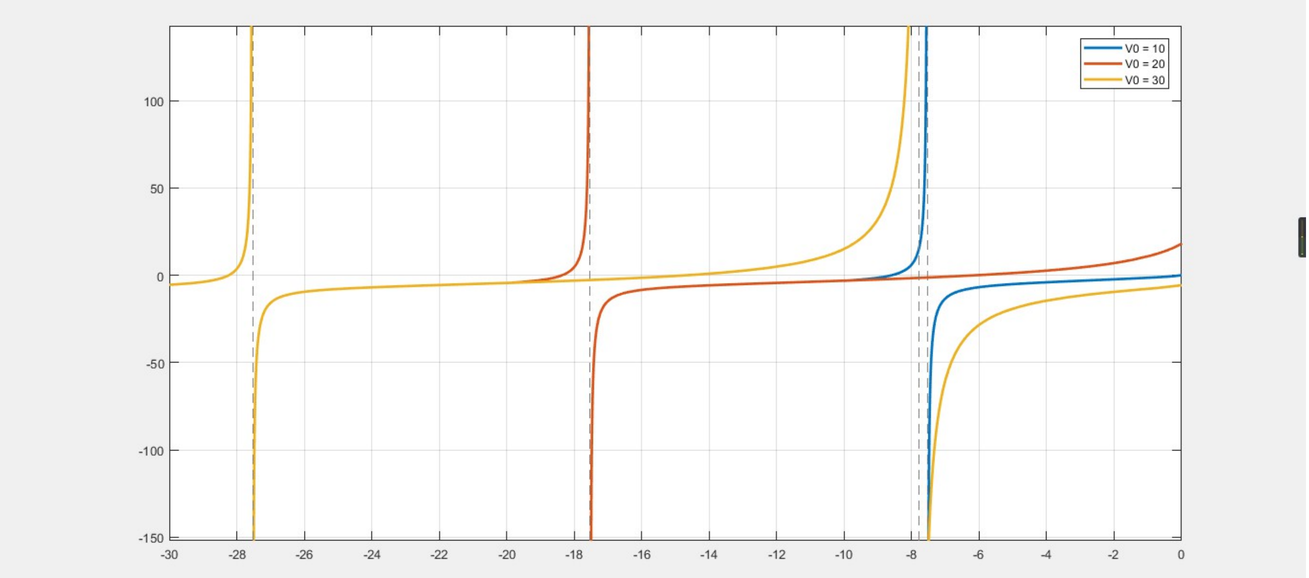
```
>> f_even = @(x) sqrt(10+x).*tan(sqrt(10+x))-sqrt(-x);
>> bisection(f_even, -10, -8);
Bisec: The root -8.5927852753 was found after 35 iterations
>>
```

And the root is **-8.5927852753**

(b)

1. For even wave function:

First we deal with deeper potential suitation, by changing V_0 from 10 to 20 and 30, plot these three lines in turn:



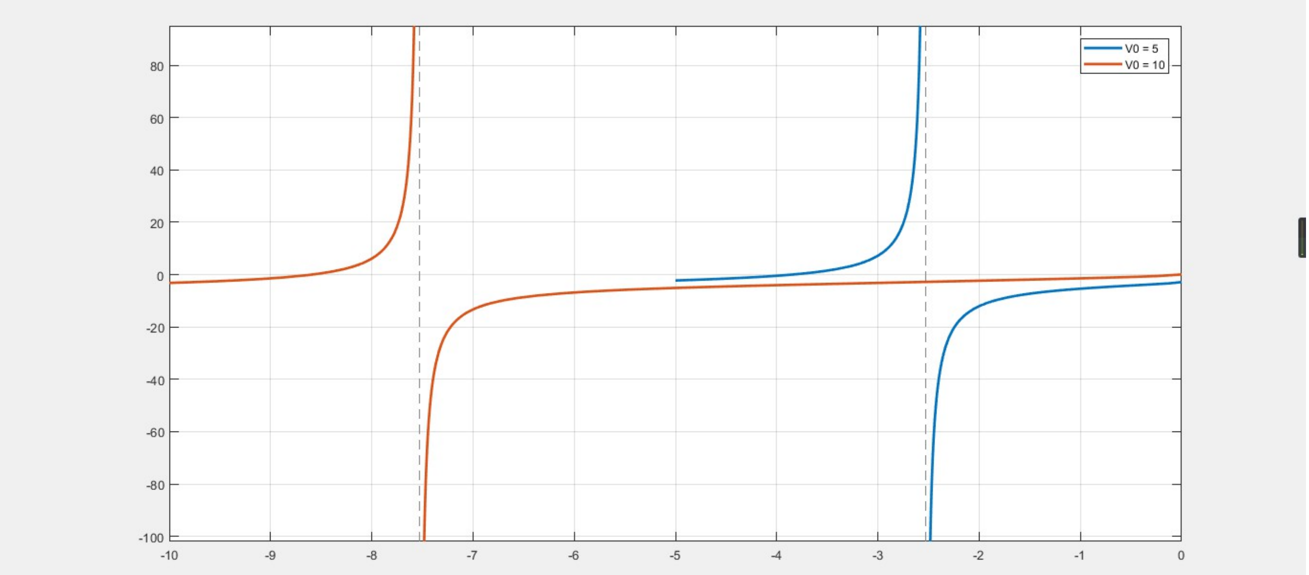
We can see, for $V_0 = 10$, we have already discussed before; for $V_0 = 20$, we have two roots, one is between -20 and -18, another is between -8 and -6; for $V_0 = 30$, we also have two roots, one is between -30 and -28, another is between -16 and -14. We use bisection fuction for each V_0 situation, code shows as follow:

```
>> f_even = @(x) sqrt(10+x).*tan(sqrt(10+x)) - sqrt(-x);
>> bisection(f_even, -10, -8);
Bisec: The root -8.5927852753 was found after 35 iterations
>> f_even = @(x) sqrt(20+x).*tan(sqrt(20+x)) - sqrt(-x);
>> bisection(f_even, -20, -18);
Bisec: The root -18.3605198524 was found after 35 iterations
>> bisection(f_even, -8, -6);
Bisec: The root -6.1084670176 was found after 35 iterations
>> f_even = @(x) sqrt(30+x).*tan(sqrt(30+x)) - sqrt(-x);
>> bisection(f_even, -30, -28);
Bisec: The root -28.2411134879 was found after 35 iterations
>> bisection(f_even, -16, -14);
Bisec: The root -14.6660534242 was found after 35 iterations
```

and display results in a graphical format:

V0	EvenRoot1	EvenRoot2
10	-8.5927852753	None
20	-18.3605198524	-6.1084670176
30	-28.2411134879	-14.6660534242

Second we deal with shallower potential suitation, we simply chage V_0 from 10 to 5, the picture shows as follow:



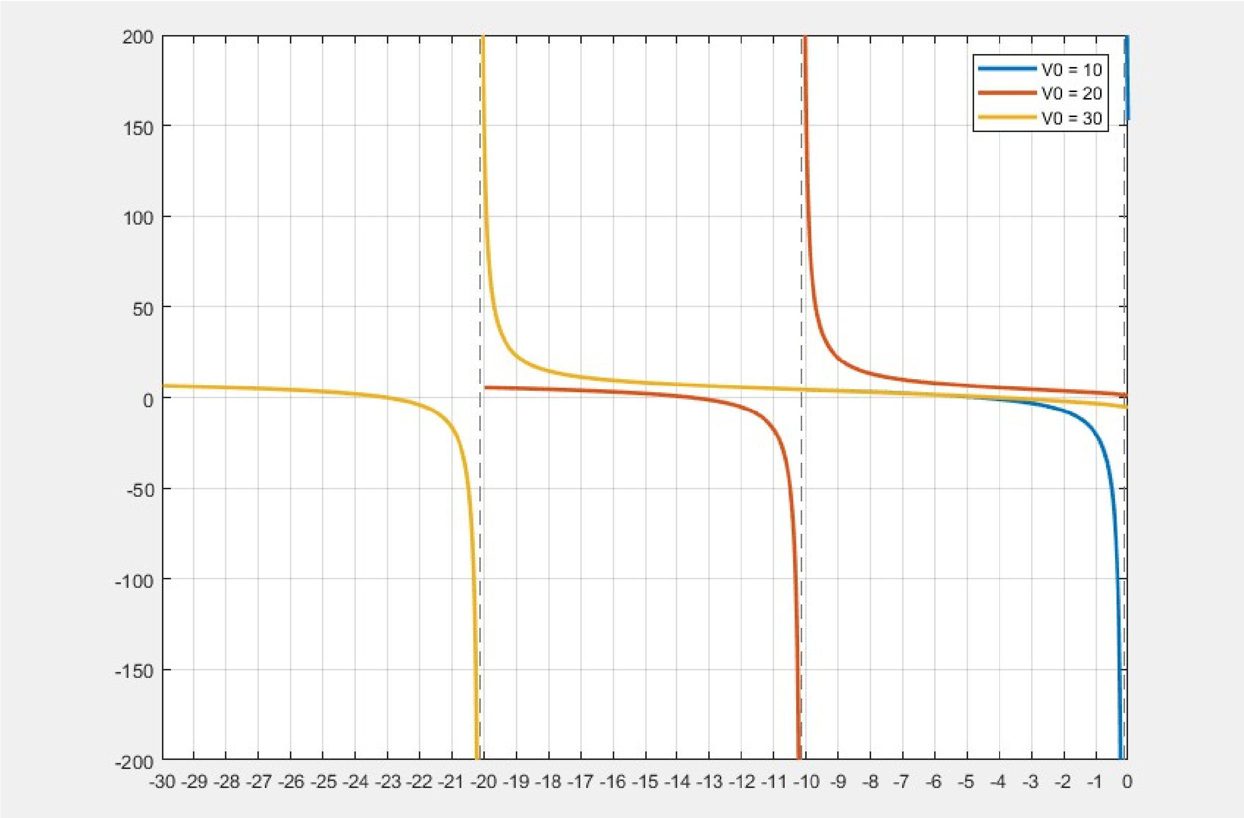
for $V_0 = 5$, there is a root between -5 and -3, by using bisection function, we can find the root = -3.8525046254, combine it with the result above:

V0	EvenRoot1	EvenRoot2
5	-3.8525046254	None
10	-8.5927852753	None
20	-18.3605198524	-6.1084670176
30	-28.2411134879	-14.6660534242

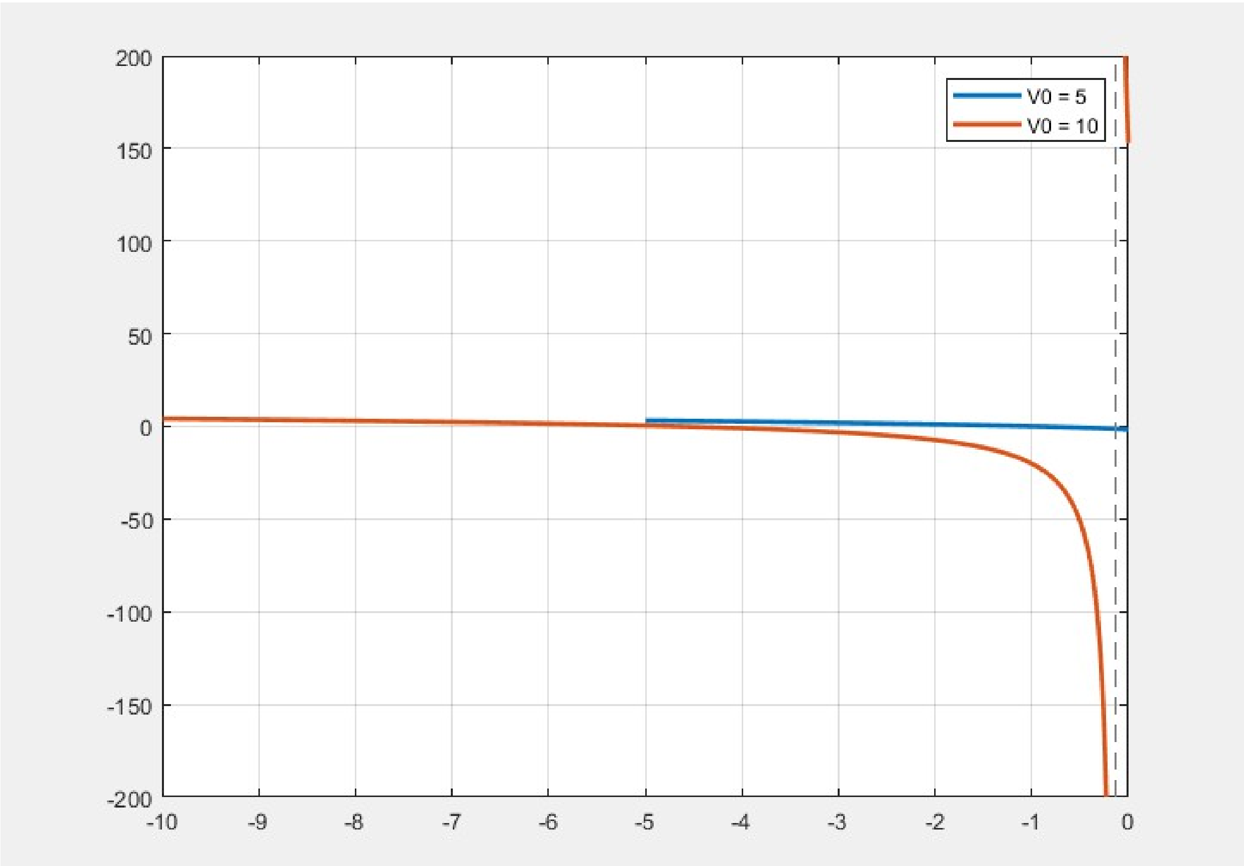
2. For odd wave function:

We simply change the even function to odd function: $f(x) = \sqrt{10+x} * cot(\sqrt{10+x}) + \sqrt{-x}$ and then repeat the previous steps, picture shows as follow:

- Deeper potential suitation



- Shallower potential suitation



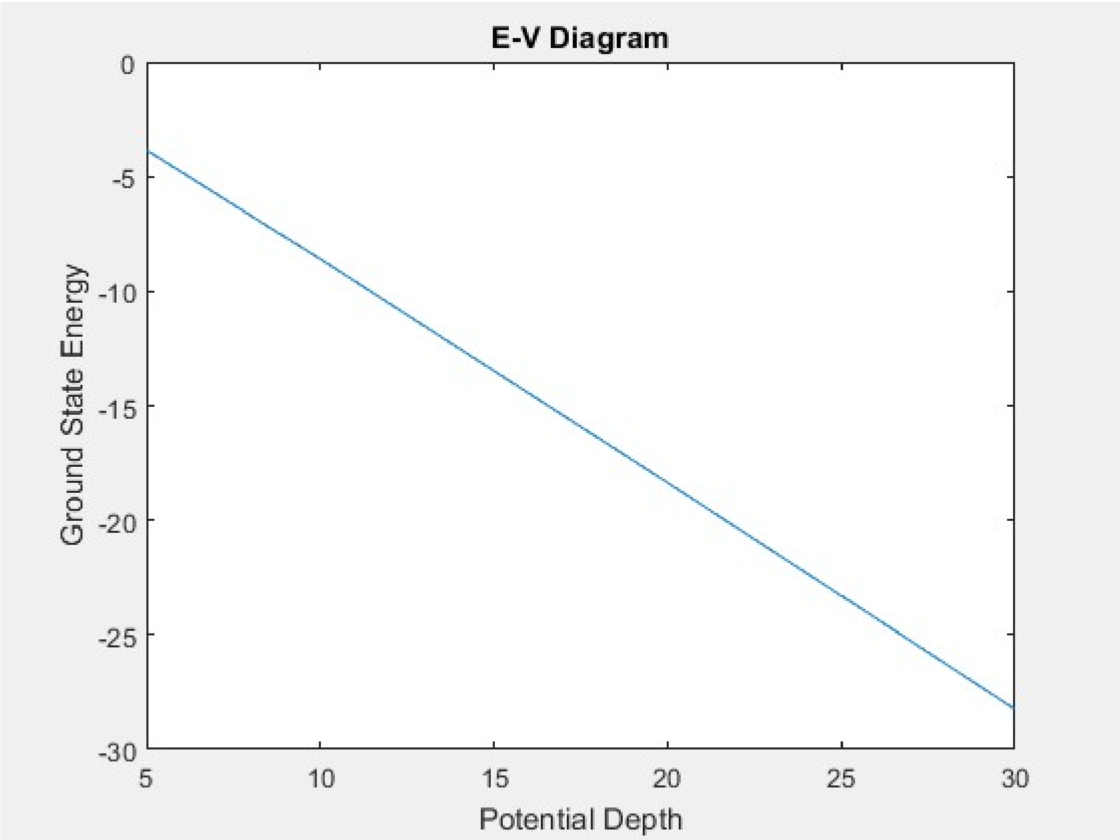
After running the bisection function, the result shows as follow:

V0	OddRoot1	OddRoot2
5	-0.9314261195	None
10	-4.6241940863	None
20	-13.5581200428	None
30	-23.0362476393	-4.0873680211

To sum up, we only choose the ground state energy:

V0	GroundEnergy
5	-3.8525046254
10	-8.5927852753
20	-18.3605198524
30	-28.2411134879

Then use matlab to plot E-V Diagram:



(c)

The transcendental equations:

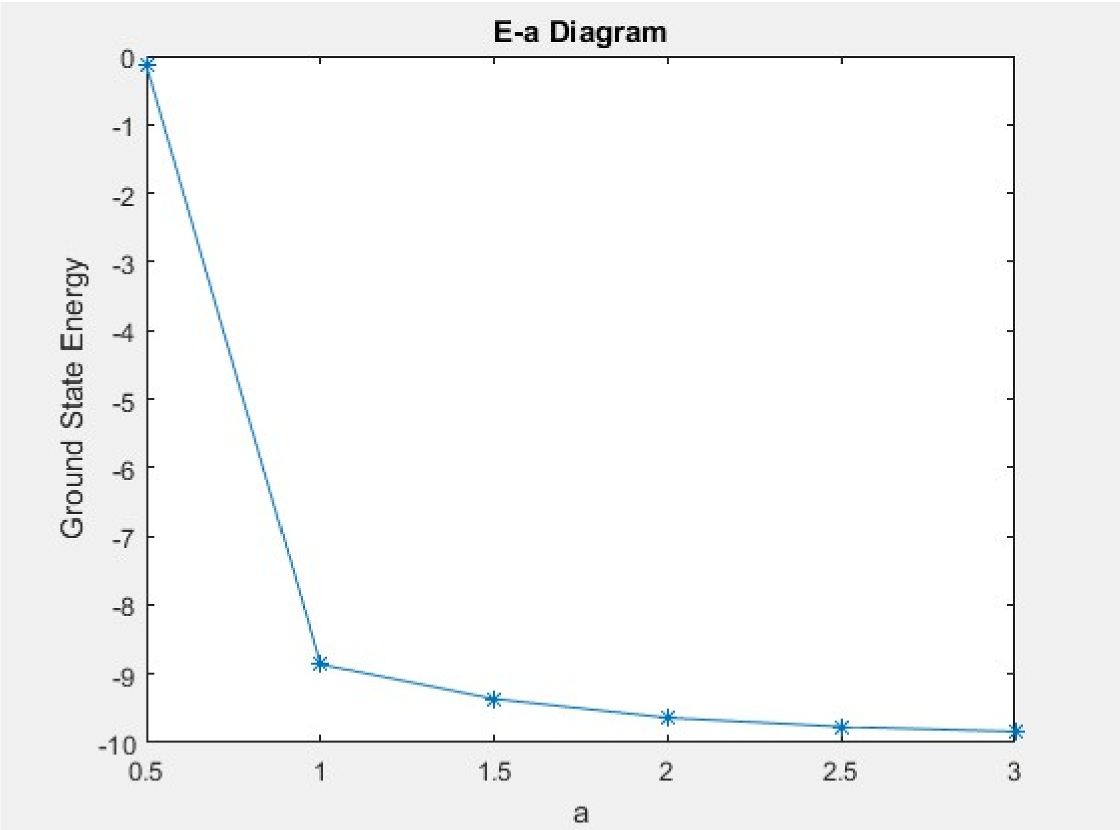
$$a\sqrt{10+E} * tan(a\sqrt{10+E}) = \sqrt{-E} \quad (even)$$

$$a\sqrt{10+E} * cot(a\sqrt{10+E}) = -\sqrt{-E} \quad (odd)$$

We choose a = 0.5, 1, 1.5, 2, 2.5, 3, for each a, we repeat the same steps in (b), and result shows as follow:

a	EvenRoot	OddRoot	GroundEnergy
0.5	-0.1303955989	None	-0.1303955989
1.0	-8.5927852753	-4.6241940864	-8.5927852753
1.5	-9.3624507400	-7.4064889139	-9.3624507400
2.0	-9.6390756815	-8.5065840410	-9.6390756815
2.5	-9.7683355015	-9.0345548958	-9.7683355015
3.0	-9.8388697163	-9.3260104982	-9.8388697163

Use matlab to plot a-GroundEnergy diagram as follow:



3. Find the bond length of NaCl

Solution:

The bond length r is the equilibrium distance when $V(r)$ is at its minimum, therefore we search for the root of $f(x) = dg(x)/dx = 0$, with $g(x) = V(x)$, the concrete form shows as follow:

$f(r) = \frac{dg(r)}{dr} = \frac{e^2}{r^2} - \frac{V_0}{r_0} e^{-\frac{r}{r_0}}$

Here we use the Newton-Method to find the root of $f(x) = 0$, and the code shows as follow:

```
clc;
clear;
% 初始化参数
x1 = 2.5;           % 初始点
e_square = 14.4;    % e^2
v0 = 1.09e3;        % V0
r0 = 0.330;         % r0
v = @(x) -e_square / x + v0 * exp(-x / r0);      % V(r)函数
f = @(x) e_square / x^2 - v0 / r0 * exp(-x / r0);       % V(r)函数的导数形式

% 牛顿法
Newton(f, x1);

% 内置函数结果
res = fminsearch(v, x1);
fprintf("The answer calculated by built-in function is %.10f\n", res);
```

CodeName:**bondLength.m**

After running, we can get the result as follow:

```
N:1      x1:2.5000000000     x2:2.3143924119
N:2      x1:2.3143924119     x2:2.3568535735
N:3      x1:2.3568535735     x2:2.3605134200
N:4      x1:2.3605134200     x2:2.3605384830
N:5      x1:2.3605384830     x2:2.3605384842
N:6      x1:2.3605384842     x2:2.3605384842
Newton: The root 2.3605384842 was found after 6 iterations.
The answer calculated by built-in function is 2.3605346680
>>
```

The results of these two methods are the same, both are **2.36053**

4. The two roots

Solution:

(a)

Use matlab to plot g1-x、g2-x、x-x, code writes as follow:

```
clc;
clear;
% 初始化参数
x = -6:0.001:4;

g1 = -3 * log2(2.2 - exp(x));
```

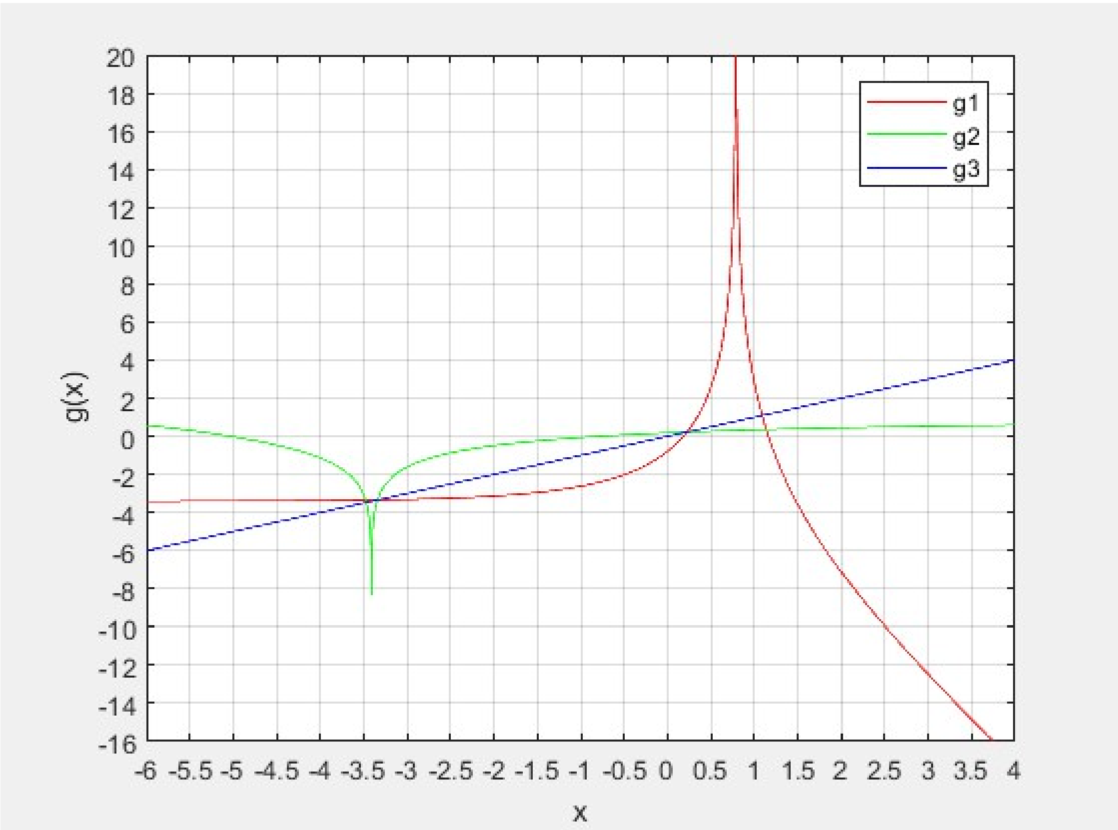
```
g2 = log(2.2 - 2.^(-x/3));
g3 = x;

plot(x, g1,'color','r');     % 画出第一条线，颜色为红
hold on;
plot(x, g2,'color','g');     % 画出第二条线，颜色为绿
hold on;
plot(x, g3, 'color', 'b');  % 画出第三条线，颜色为蓝

xlabel('x');          % 横坐标为x
ylabel('g(x)');       % 纵坐标为g(x)
axis([-6 4 -16 20]);     % 设定横坐标范围为-6到4，纵坐标范围为-16到20
set(gca, 'XTick', -6:0.5:4);     % 横坐标间隔为0.5
set(gca, 'YTick', -16:2:20);     % 纵坐标间隔为2
legend("g1", "g2", "g3");        % 给三条线标注
grid on;     % 显示网格线
```

CodeName:**Draw.m**

the picture shows as follow:



Note:

> $g_1(x) = -3log_x(2.2 - e^x)\ g_2(x) = ln(2.2 - 2^{-x/3})\ g_3(x) = x$

As we can see in the picture, the first fixed point nears **-3.5**, and the second fixed point nears **0.25**

(b)

According to (a), we choose initial value of x1 as -4 and -3, by using fixed point method, we write code as follow:

```
clc;
clear;
% 初始化参数
init = [-4 -3];   % 初始点
g1 = @(x) -3 * log2(2.2 - exp(x));      % 函数g1(x)
g2 = @(x) log(2.2 - 2.^(-x/3));         % 函数g2(x)

for i = 1:2     % 有两个函数，所以加两个循环
    if i == 1
        f  = g1;
    else
        f = g2;
    end
    for j = 1:2     % 有两个初始点，所以加两次循环
        x1 = init(1, j);
        fprintf("Function: g%d(x) Initial Value: %d\n", i, x1);
        FixedPoint(f, x1);      % 为了便于显示，这里将FixedPoint函数中的详细输出注释掉了
    end
end
```

CodeName:**FindRoot.m**

Note:

> The question requires only four iterations, so remember to change the N in FixedPoint.m to 4

After running the code, we can get the result as follow:

```
Function: g1(x) Initial Value: -4
N:1       x1:-4.0000000000     x2:-3.3763271772
N:2       x1:-3.3763271772     x2:-3.3447545574
N:3       x1:-3.3447545574     x2:-3.3425635381
N:4       x1:-3.3425635381     x2:-3.3424088656
Function: g1(x) Initial Value: -3
N:1       x1:-3.0000000000     x2:-3.3134386030
N:2       x1:-3.3134386030     x2:-3.3403197620
N:3       x1:-3.3403197620     x2:-3.3422501114
N:4       x1:-3.3422501114     x2:-3.3423867114
Function: g2(x) Initial Value: -4
N:1       x1:-4.0000000000     x2:-1.1399278431
N:2       x1:-1.1399278431     x2:0.4056749224
N:3       x1:0.4056749224      x2:0.2661725330
N:4       x1:0.2661725330      x2:0.2311698895
Function: g2(x) Initial Value: -3
N:1       x1:-3.0000000000     x2:-1.6094379124
N:2       x1:-1.6094379124     x2:-0.2882537376
N:3       x1:-0.2882537376     x2:0.1232183242
N:4       x1:0.1232183242      x2:0.2054422362
fx >>
```

As we can see in the result, fixed point is not found both $g_1(x)$ and $g_2(x)$, no matter initial value is -4 or -3. And for $g_1(x)$, x1 converges to -3.34 while for $g_2(x)$, x1 converges to 0.2, so the different slopes converge to different points.

5. Halley's method

Solution

(a)

Assume r is a fixed point of $f(x)$, and r is near to x_n, according to Taylor's theorom:

$0 = f(r) = f(x_n) + f'(x_n)(r - x_n) + \frac{f''(\eta)}{2}(r - x_n)^2$

$0 = f(r) = f(x_n) + f'(x_n)(r - x_n) + \frac{f''(x_n)}{2}(r - x_n)^2 + \frac{f'''(\xi)}{6}(r - x_n)^3$

multiply the second equation by $2f'(x_n)$ and subtract it from the first equation times $f''(x_n)(a - x_n)$:

$0 = 2f'(x_n)f(x_n) + 2[f'(x_n)]^2(r - x_n) + f'(x_n)f''(x_n)(r - x_n)^2 + \frac{f'(x_n)f'''(\xi)}{3}(r - x_n)^3 - f(x_n)f''(x_n)(r - x_n) - f'(x_n)f''(x_n)(r - x_n)^2 - \frac{f''(x_n)f''(\eta)}{2}(r - x_n)^3$

After simplification:

$0 = 2f'(x_n)f(x_n) + (2[f'(x_n)]^2 - f(x_n)f''(x_n))(a - x_n) + (\frac{f'(x_n)f'''(\xi)}{3} - \frac{f''(x_n)f''(\eta)}{2})(r - x_n)^3$

According to Halley's Method:

$$x_{n+1} = x_n - \frac{2f(x_n)f'(x_n)}{2[f'(x_n)]^2 - f(x_n)f''(x_n)}$$

We can get:

$$r - x_{n+1} = -\frac{2f'(x_n)f'''(\xi) - 3f''(x_n)f''(\eta)}{6(2[f'(x_n)]^2 - f(x_n)f''(x_n))}(r - x_n)^3$$

let $x_n \to r$, we have:

$$\Delta x_{n+1} = \frac{3(f'')^2 - 2f'f'''}{12(f')^2}(\Delta x_n)^3 + \mathcal{O}([\Delta x_n]^4)$$

As we can see, Halley's method has order of convergence 3.

(b)

Accoring to Healley's method, write code as follow:

```
function Helley(f, x1)
syms x;
tol = 1e-10;                 % 容许误差
diff_f_1 = diff(f(x));       % f函数的一阶导数
diff_f_2 = diff(f(x), 2);    % f函数的二阶导数
N = 100;                     % 默认循环次数
for n = 1:N
    x = x1;
    tmp_f_1 = eval(diff_f_1);% 一阶导数的具体数值
    tmp_f_2 = eval(diff_f_2);% 二阶导数的具体数值
    x2 = x1 - 2*f(x1)*tmp_f_1/(2*(tmp_f_1).^2 - f(x1)*tmp_f_2);
    fprintf("N:%d \t x1:%.10f \t x2:%.10f\n", n, x1, x2);
    if abs(x2 - x1) < tol
        r = x2;
        fprintf("Helley: The root %.10f was found after %d iterations\n", r, n);
        break;
    end
    x1 = x2;
end
```

CodeName:**Helley.m**

Running this function, for $f(x) = 5x^7 + 2x - 1$:

```
>> f = @(x) 5*x.^7 + 2*x - 1;
>> Helley(f, 1.5)
N:1       x1:1.5000000000      x2:1.1143125486
N:2       x1:1.1143125486      x2:0.8005353482
N:3       x1:0.8005353482      x2:0.5300709040
N:4       x1:0.5300709040      x2:0.4841182191
N:5       x1:0.4841182191      x2:0.4843634906
N:6       x1:0.4843634906      x2:0.4843634906
Helley: The root 0.4843634906 was found after 6 iterations
```

and for $g(x) = 1/x^3 - 10$:

```
>> f = @(x) 1./x.^3 - 10;
>> Helley(f, 1.5)
N:1       x1:1.5000000000     x2:0.7828467153
N:2       x1:0.7828467153     x2:0.5022523157
N:3       x1:0.5022523157     x2:0.4643100491
N:4       x1:0.4643100491     x2:0.4641588834
N:5       x1:0.4641588834     x2:0.4641588834
Helley: The root 0.4641588834 was found after 5 iterations
```

(c)

For bisection method, we call the bisection function defined before, for $f(x) = 5x^7 + 2x - 1$:

```
>> f = @(x) 5*x.^7 + 2*x - 1;
>> bisection(f, -2, 2)
Bisec: The root 0.4843634905 was found after 36 iterations
fx >>
```

and for $g(x) = 1/x^3 - 10$:

```
>> f = @(x) 1./x.^3 - 10;
>> bisection(f, -2, 2)
There is no root between -2.000000 and 2.000000
>> bisection(f, -1, 1)
There is no root between -1.000000 and 1.000000
>> bisection(f, 0, 1)
Bisec: The root 0.4641588834 was found after 34 iterations
fx >>
```

For Newton's method, we also call the Newton function defined before, for $f(x) = 5x^7 + 2x - 1$:

```
>> f = @(x) 5*x.^7 + 2*x - 1;
>> Newton(f, 1.5)
N:1       x1:1.5000000000     x2:1.2817923020
N:2       x1:1.2817923020     x2:1.0910637579
N:3       x1:1.0910637579     x2:0.9209388003
N:4       x1:0.9209388003     x2:0.7645930108
N:5       x1:0.7645930108     x2:0.6208120730
N:6       x1:0.6208120730     x2:0.5160770920
N:7       x1:0.5160770920     x2:0.4856762021
N:8       x1:0.4856762021     x2:0.4843654698
N:9       x1:0.4843654698     x2:0.4843634906
N:10      x1:0.4843634906     x2:0.4843634906
Newton: The root 0.4843634906 was found after 10 iterations.
```

and for $g(x) = 1/x^3 - 10$:

```
>> f = @(x) 1./x.^3 - 10;
>> Newton(f, 0.5)
N:1      x1:0.5000000000     x2:0.4583333333
N:2      x1:0.4583333333     x2:0.4640138728
N:3      x1:0.4640138728     x2:0.4641587928
N:4      x1:0.4641587928     x2:0.4641588834
N:5      x1:0.4641588834     x2:0.4641588834
Newton: The root 0.4641588834 was found after 5 iterations.
```

Reference:

1. https://en.wikipedia.org/wiki/Halley%27s_method
2. Numerical Methods for Engineers and Scientists Using MATLAB, Ramin S. Esfandiari