Q1.

code to compute the result are as followings

```
function bisectionAns = bisection(a,b)
    format long;
    mid = b;
    while abs(compute(mid))>0.00001                                %set accuracy
        mid = (a+b)/2;
        if compute(mid)*compute(b)>0
            b = mid;
        else
            a = mid;
        end
    end
    bisectionAns = b;
end

function secantAns = secant(a,b)
    format long;
    next = b;
    while abs(compute(next))>0.00001
        next = b-compute(b)*(b-a)/(compute(b)-compute(a));  % "next" is the intersection of the line with point a & b and y=0
        if compute(next)*compute(b)>0                        % let the true solution always in [a,b]
            b = next;
        else
            a = next;
        end
    end
    secantAns = next;
end

function newtonAns = newton(a)
    format long;
    next = a;
    while abs(compute(next))>0.00001
        next = next - compute(next)/computeD(next);          % x(n+1) = x(n) - f(xn)/f'(xn)
    end
    newtonAns = next;
end
function computeAns = compute(m)                              % compute f(x)
    computeAns = power(m,2) + sin(m) - exp(m)/4 - 1;
end

function computeDAns = computeD(input)                        % compute f'(x)
    computeDAns = 2*input + cos(input) + exp(input)/4;
end
```

the result

```
>> Q1
bisection : between -2 & 0
  -1.431808471679688

bisection : between 0 & 2
   0.911918640136719

secant : between -2 & 0
  -1.431806711938404

secant : between 0 & 2
   0.911916728109482

Newton's : between -2 & 0, starting from -1
  -1.431807288773363

Newton's : between 0 & 2, starting from 1
   0.911922415462871
```

Q2.

change the function f(x) and f'(x) and use Newton's as in Q1

```
format long
ans1 = newton(3);
disp('ans :');
disp(ans1);

function newtonAns = newton(a)
    format long;
    next = a;
    while abs(compute(next))>0.00001
        next = next-compute(next)/computeD(next);
    end
    newtonAns = next;
end

function computeDAns = computeD(x)          % compute f'(x)
    format long;
    computeDAns = power(x-2,2)*(x-4)*(5*x-16);
end

function computeAns = compute(x)            % compute f(x)
    format long;
    computeAns = power(x-2,3)*power(x-4,2);
end
```

the result

```
>> Q2
ans :
     2
```

If we start from x0 = 3, it will only take 1 step to reach the solution. However, since there are multiple roots on either of the solution x = 2 and x = 4, f'(R) = 0, so the g'(R) in the induction process can't be omitted, and therefor the convergence was linear instead of quadratic. The convergence will be quadratic only if we modify the newton's method into "x(n+1) = x(n) – (number of roots in a particular solution x) * f(x)/f'(x)".

Q3.

first compute f(x), pick the easiest one x = sqrt(4/x)

$$x = \sqrt{\frac{4}{x}}$$

$$x^2 = \frac{4}{x}$$

$$x^3 - 4 = 0 = f(x)$$

$$f(x) = x^3 - 4$$

Then I write a fixed-point iteration method code to compute.
When iterating more than 1000000 times, I consider it to be diverge.

```
function fixedAns = fixed(a,b)
    format long;
    iteration = 0;
    arrayForLoop = ones(100);
    loop = 0;
    for i=1:100
        arrayForLoop(i)=inf;
    end
    while abs(compute(a))>0.000001
        a = g(a,b);
        if iteration>1000000              % set limit of iterations
            fixedAns = 'diverge(too many iteration)';
            break;
        elseif a==inf
            fixedAns = "diverge(inf)";
            break;
        end
        for i=1:100
            if a==arrayForLoop(i)
                fixedAns = "diverge(loop)";
                loop = 1;
                break;
            end
        end
        arrayForLoop(mod(iteration,100)+1) = a;
        iteration = iteration+1;
    end
    if iteration<=1000000 && a~=inf && ~loop
        fixedAns = a;
    end
end

function computeAns = compute(a)          % compute f(x)
    format long;
    computeAns = power(a,3)-4;
end
function gAns = g(a,b)                    % 3 different g(x)
    format long;
    if b == 1
        gAns = (4+2*power(a,3))/power(a,2)-2*a;
        %disp(gAns);
    elseif b == 2
        gAns = sqrt(4/a);
    else
        gAns = (16+power(a,3))/(5*power(a,2));
    end
end
```

By some simple test, I roughly predict that choice (a)(c) will diverge, and choice (b) will converge.

Then we try to prove it.

converge if $\rho(J(R)) < 1$

we know root $x^* \approx 2^{\frac{2}{3}}$ by the test using (b)

(a)

$$J = \left[\frac{g(x)}{dx}\right]$$

$$\frac{g(x)}{dx} = \frac{6x^4 - 8x - 4x^4}{x^4} - 2 = \frac{-8}{x^3},\ x = 2^{\frac{2}{3}},\ \lambda = -1$$

$\lambda = -1 \rightarrow \rho(J(R)) \geq 1 \rightarrow$ diverge

(if there are other root)

(b)

$$\frac{g(x)}{dx} = -x^{-\frac{3}{2}},\quad J(2^{\frac{3}{2}}) = \left[2^{-\frac{9}{4}}\right]$$

$\lambda = -2^{-\frac{9}{4}} < 1 \rightarrow \rho(J(R)) < 1 \rightarrow$ converge

(c)

$$\frac{g(x)}{dx} = \frac{3x^2 \cdot 5x^2 - (16 + x^3) 10x}{25x^4} = \frac{5x^4 - 160x}{25x^4} = \frac{1}{5} - \frac{32}{5x^3}$$

$x = 2^{\frac{2}{3}},\ \frac{g(x)}{dx} = -\frac{14}{5} < -1$

$\sqrt{\lambda^2} > 1 \rightarrow \rho(J(R)) > 1 \rightarrow$ diverge

As for the range of convergence for (b)

we know $e_k = x_k - x^*$, with $g(x^*) = x^*$ and Taylor's theorem

$\Rightarrow e_{k+1} \approx g'(x^*) e_k$

It will converge when $e_{k+1} < e_k \rightarrow |g'(x^*)| < 1$

when given (a, b) which $x^* \in (a, b)$

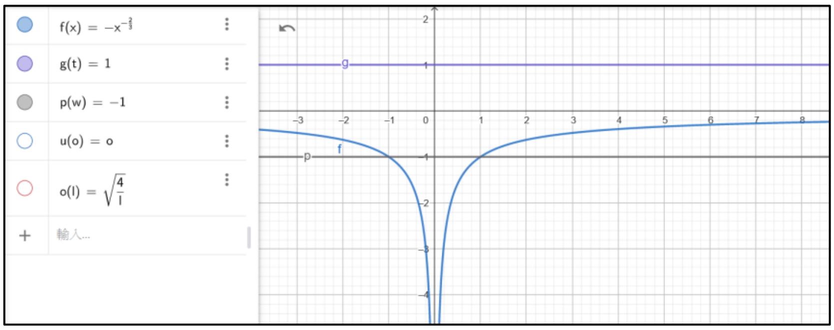
if $\exists k < 1$ s.t. $|g'(x)| < k \quad \forall x \in (a, b)$,

then $\forall x_0 \in (a, b)$ will converge to x^*

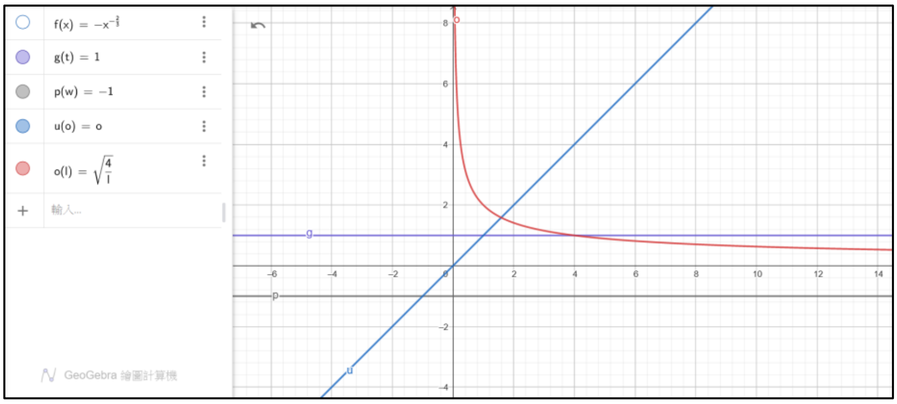
(b)

$g'(x) = -x^{-\frac{3}{2}}$

For (b), we can see that in (1.01, inf), we can find k = (1+g'(1.01))/2, so for all x0 in (1.01, inf) ,x0 will converge to x*.



After we know choice (b) will converge when x0 ∈ (1.01, inf), we can know that choice (b) will also converge when x0 ∈ (0, 1.01]. When choosing x0 ∈ (0, 1.01], x1 will ∈ (1.01, inf] (this can be known easily by looking the graph), so choice (b) will converge when x0 ∈ (0, inf).



Then I test the convergence for negative values with the following code.

```
testx = linspace(-9000,-0.000001,100000);
corr = 0;
for i = 1:100000
    ans1=fixed(testx(i),2);
    if class(ans1)=="double"
        if abs(compute(ans1))<0.000001
            corr = corr+1;
        end
    end
end
disp(corr);
```

The result shows that choice (b) also converge for all the negative point I chose.

```
>> Q3
       100000
```

If we don’t take imaginary number calculation into account, the choice (b) will converge if x0 ∈ (0, inf), but if we do, it might converge if x0 ∈ R – {0}.

Q4.

Solve with the following codes

```

format long
x = input("x:"); y = input("y:");z = input("z:");
[x,y,z] = newton3(x,y,z);
disp("ans: ")
disp([x y z])
disp("error")
disp([x-3*y-z^2+3  2*x^3+y-z^2*5+2 4*x^2+y+z-7])

function [x, y, z] = newton3(x, y, z)
    for N = 1:100000
        D = [1,-3,-2*z;6*x^2,1,-10*z;8*x,1,1];
        f1 = x-3*y-z^2+3 ;
        f2 = 2*x^3+y-z^2*5+2 ;
        f3 = 4*x^2+y+z-7;
        s = [x y z]' ;
        s = s - D\[f1 f2 f3]' ;
        x = s(1);
        y = s(2);
        z = s(3);
    end
end
```

Here is the result (ans[x y z] & error)

```
ans:
   1.111408182587243   0.988209723132635   1.070877683579846

error
   1.0e-15 *

                   0  -0.888178419700125                   0
```

```
ans:
   1.353748286168190   0.925430625881817  -1.255968315095066

error
   1.0e-15 *

                   0  -0.888178419700125                   0
```

```
ans:
   1.0e+03 *

   0.031151404846176  -3.768157126110150  -0.106482969451348

error
   1.0e-11 *

                   0  -0.727595761418342   0.019895196601283
```

```
ans:
   1.0e+03 *

   0.032884630662930  -4.434086620233489   0.115490884884321

error
     0     0     0
```

This 4 results can be found easily by running the program I wrote which choose x,y,z randomly.

I also tried fixed point iteration with three different ways, but still couldn't find any result other than this four.

Then I use the function built in matlab

```
x0 = [800, 90, 900]; % initial guess
x = fsolve(@myfun, x0);
disp(x)
function F = myfun(x)
    F = [x(1)-3*x(2)-x(3)^2+3;
         2*x(1)^3 + x(2) - 5*x(3)^2+2;
         4*x(1)^2 + x(2) + x(3)-7];
end
```

Here are the 2 remaining results

```
  Columns 1 through 2

 -1.250595988930628 + 0.048994659662924i   0.665052997312710 - 0.013409657718146i

  Column 3

  0.088587599272409 + 0.503589856545989i

>> x(1)-3*x(2)-x(3)^2+3

ans =

     2.401190357659289e-12 - 1.330394128196133e-12i

>> 2*x(1)^3 + x(2) - 5*x(3)^2+2

ans =

     1.074118571864346e-11 - 6.655231921115501e-12i

>> 4*x(1)^2 + x(2) + x(3)-7

ans =

     6.821210263296962e-13 + 5.273559366969494e-14i
```

```
 -1.250595988916906 - 0.048994659667079i   0.665052997308376 + 0.013409657718614i

  Column 3

  0.088587599418753 - 0.503589856582128i

>> x(1)-3*x(2)-x(3)^2+3

ans =

     3.959765848549068e-11 + 1.495692458775011e-10i

>> 2*x(1)^3 + x(2) - 5*x(3)^2+2

ans =

     1.903806001735120e-10 + 7.472717999945644e-10i

>> 4*x(1)^2 + x(2) + x(3)-7

ans =

     3.778310997404333e-12 + 4.661826480401032e-13i
```

Top -> the answer [x, y, z]

The following 3 line -> verify the accuracy.