Root Finding

wiki: https://en.wikipedia.org/wiki/Root-finding_algorithm

generate data

```
hf1 = @sin;
hf1_gradient = @cos;
a = -rand();
b = rand();
x0 = 1.1;
```

built-in

```
ret1 = fsolve(hf1,x0);
```

```
Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the default value of the function tolerance, and
the problem appears regular as measured by the gradient.

<stopping criteria details>
```

```
disp(ret1)
```

```
    1.2495e-11
```

Bisection method

wiki: https://en.wikipedia.org/wiki/Bisection_method

linear convergence: $\epsilon_{n+1} = \epsilon_n/2$

```
ret1 = my_bisection(hf1, a, b, 20);
disp(ret1)
```

```
    2.7277e-07
```

Newton-Raphson method

wiki: https://en.wikipedia.org/wiki/Newton%27s_method

$$x_{n+1} = x_n - \frac{f(x)}{f'(x)}$$

quadratic convergence: $\epsilon_{n+1} = \frac{|f''(\xi_n)|}{2|f'(\xi_n)|}\epsilon_n^2$

```
ret1 = my_newton_raphson(hf1, hf1_gradient, x0, 10);
disp(ret1)
```

```
     0
```

function

```
function [ret,tracks] = my_bisection(hf1, a, b, step)
% find root using bisection method
% reference: https://en.wikipedia.org/wiki/Bisection_method
% hf1(function handle), a(float), b(float) step(int)
% ret(float), tracks(1,step)
fa = hf1(a);
fb = hf1(b);
assert(fa*fb<0, 'bisection method require function value signs at a and b differ');
tracks = zeros(1,step);
for ind1 = 1:step
    new_x = (a+b)/2;
    fx = hf1(new_x);
    tracks(ind1) = new_x;
    if fx*fa<0
        fb = fx;
        b = new_x;
    elseif fx*fb<0
        fa = fx;
        a = new_x;
    else
        break;
    end
end
tracks = tracks(1:ind1);
ret = tracks(ind1);
end

function [ret,tracks] = my_newton_raphson(hf1, hf1_gradient, x0, step)
% find root using Newton-Raphson method
% reference: https://en.wikipedia.org/wiki/Newton%27s_method
% hf1(function handle)
% hf1_gradient(function handle) gradient of function hf1
% step(int)
% ret(float), tracks(1,step)
tracks = zeros(1,step);
tracks(1) = x0;
for ind1 = 2:step
    x0 = x0 - hf1(x0)/hf1_gradient(x0);
    tracks(ind1) = x0;
end
ret = tracks(end);
end
```