A bungee jumper’s velocity as a function of time can be described by the equation,

\[ v(t) = \frac{gm}{c_d} \tanh \left( \frac{gc_d}{m} t \right), \]

where \( v(t) \) is the time-dependent velocity, \( g \) is the gravitational constant for earth, 9.81 m/s\(^2\), \( m \) is the jumper’s mass, 65 kg, and \( c_d \) is the drag coefficient.

1. Plot the jumper’s velocity at time 4.5 s, as a function of the drag coefficient for a velocity range, 20 m/s < \( v(t) \) < 40 m/s.

2. If the jumper has a velocity of 35 m/s after a time of 4.5 s, determine her coefficient of drag using,
   a. The bisection method, solved by hand using MATLAB to an accuracy of
      \[ \frac{c_{d,u} - c_{d,l}}{c_{d,u}} < 0.1. \]
   b. The bisection method, using a program (m-file) to an accuracy of
      \[ \frac{c_{d,u} - c_{d,l}}{c_{d,u}} < 0.0001. \]
   c. The false-position method, solved by hand using MATLAB to an accuracy of
      \[ \frac{c_{d,u} - c_{d,l}}{c_{d,u}} < 0.1. \]
   d. The false-position method, using a program (m-file) to an accuracy of
      \[ \frac{c_{d,u} - c_{d,l}}{c_{d,u}} < 0.0001. \]
   e. Compare the convergence rates of the methods.
Problem 1.

We are trying to determine the range of \( cd \) for \( 20 < v(t) < 40 \). First just get a range of \( cd \) in which the range of \( cd \) lies. Look at the plot, and determine the correct range of \( cd \)...

\[
\begin{align*}
&\text{>> cd = linspace(.1, 2); } \\
&\text{>> g = 9.81; } \\
&\text{>> m = 65; } \\
&\text{>> v = sqrt(g*m./cd).*tanh(sqrt(g*cd*t/m)); } \\
&\text{>> plot(cd,v) } \\
\end{align*}
\]

From this plot we see that \( 0.1 < cd < 1.45 \rightarrow 20 < v < 40 \). Now plot...

\[
\begin{align*}
&\text{>> cd = linspace(.1, 1.45); } \\
&\text{>> v = sqrt(g*m./cd).*tanh(sqrt(g*cd*t/m)); } \\
&\text{>> plot(cd,v) } \\
&\text{>> grid on } \\
\end{align*}
\]
Problem 2a.

```matlab
>> v = 35;
>> fcd = @(cd)sqrt(g*m./cd).*tanh(sqrt(g*cd/m)*t)-v
fcd =
    @(cd)sqrt(g*m./cd).*tanh(sqrt(g*cd/m)*t)-v
>> cdl = 0.2; ← choose a left value of the interval
>> fcdl = fcd(cdl)
fcdl =
    1.9136
>> cdu = 0.4; ← choose a right value of the interval
>> fcdu = fcd(cdu)
fcdu =
    -2.9581
>> cdr = (cdl+cdu)/2 ← bisect the interval
cdr =
    0.3000
>> fcdr = fcd(cdr)
fcdr =
    -0.7374
>> fcdl*fcdr ← test the left interval for a root
ans =
    -1.4112 ← negative → root lies in left interval
>> cdu = cdr

cdu =
    0.3000
```
>> cdr = (cdl+cdu)/2

cdr =
    0.2500

>> fcdr = fcd(cdr)

fcdr =
    0.5267

>> fcdl*fcdr ← positive → root lies in right interval

ans =
    1.0079

>> cdl = cdr

cdl =
    0.2500

>> cdr = (cdl+cdu)/2

cdr =
    0.2750

>> fcdr = fcd(cdr)

fcdr =
    -0.1197

>> fcdl*fcdr

ans =
    -0.2290

>> cdu = cdr

cdu =
0.2750 ← upper limit of the interval

ans = 

0.0909 ← relative value of interval size

cdl = 

0.2500 ← lower limit of the interval

Problem 2b.

>> g = 9.81;
>> m = 65;
>> t = 4.5;
>> v = 35;
>> fcd = @(cd)sqrt(g*m./cd).*tanh(sqrt(g*cd/m)*t)-v

fcd = 

 @(cd)sqrt(g*m./cd).*tanh(sqrt(g*cd/m)*t)-v

>> [xr, ea, iter] = divide_and_conquer(f, xl, xu, tol)
>> xl = 0.2;
>> xu = 0.4;
>> tol = 0.00001;
>> [xr, ea, iter] = divide_and_conquer(fcd, xl, xu, tol)
xr = 

0.2703

ea = 

5.6455e-006

iter = 

17
Problem 2c.

```matlab
>> cdl = 0.2;
>> cdu = 0.4;
>> cdr = cdu - (fcd(cdu) * (cdl-cdu)) / (fcd(cdl) - fcd(cdu))

cdr =
    0.2786

>> fcd(cdl)*fcd(cdr) ← test left interval

ans =
    -0.4006 ← root is in left interval

>> cdu = cdr

cdu =
    0.2786

>> cdr = cdu - (fcd(cdu) * (cdl-cdu)) / (fcd(cdl) - fcd(cdu))

cdr =
    0.2708

>> fcd(cdl)*fcd(cdr)

ans =
    -0.0258

>> cdu = cdr

cdu =
    0.2708

>> cdr = cdu - (fcd(cdu) * (cdl-cdu)) / (fcd(cdl) - fcd(cdu))

cdr =
    0.2703
```
>> fcd(cdl)*fcd(cdr)
ans = 
   -0.0016
>> cdu = cdr
cdu = 
   0.2703
>> cdr = cdu - (fcd(cdu)*(cdl-cdu)) / (fcd(cdl)-fcd(cdu))
cdr = 
   0.2703  \leftarrow \text{Uh oh, false position method is not changing the calculation of cdr. Let’s see what’s going on…}
>> fcd(cdr)
ans =
   -5.5118e-005
>> cd = linspace(0.2, 0.4);
>> ffcd = fcd(cd);
>> plot(cd,ffcd)
>> grid on

The function is essentially a straight line. The false position method finds the value of the root, and we set cdu equal to it to eliminate the right interval. But the right interval length has gone to zero, so no progress is made!
Problem 2d.

```matlab
>> fcd = @(cd)sqrt(g*m./cd).*tanh(sqrt(g*cd/m)*t)-v
fcd =
    @(cd)sqrt(g*m./cd).*tanh(sqrt(g*cd/m)*t)-v
```

```matlab
>> %[xr, ea, iter] = divide_and_conquer(f, xl, xu, tol)
>> xl = 0.2;
>> xu = 0.4;
>> tol = 0.00001;
>> [xr, ea, iter] = divide_and_conquer(fcd, xl, xu, tol)

xr =
    0.2703

ea =
    -1

iter =
    101
```

Same deal when we use the code.