Problem: An object is shot upwards with an initial velocity of v_0 m/s. Its height h (in m) after t seconds is given by

$$h = \frac{1}{r}\left(v_0 + \frac{g}{r} \right)\left(1 - e^{-rt}\right) - \frac{gt}{r}$$

where r is a drag coefficient and g is 9.81 m/s^2.

If r = 0.35 s^{-1} and v_0 = 78 m/s

(i) When will be object be 80m above the ground?

(ii) When will the object hit the ground?

(iii) What maximum height will be attained?

A good first step in solving the problem is to implement the key equation as a function m-file (although it's admittedly on the simple side and an anonymous function could also be used).

```
function [ height ] = objectHeight( v0, r, t )
% OBJECTHEIGHT Calculates height of an object shot vertically.
% Inputs: v0 = initial velocity (in m/s)
%         r = drag coefficient (in 1/s)
%         t = time (in s), cannot be zero
% Outputs: height = height of object

if r == 0
   error ('zero drag coefficients are not allowed');
end

g = 9.81; % gravity

height = (1 / r) * (v0 + g/r) * (1 - exp(-r * t)) - g * t / r;

end
```

```
>> v0 = 78;  r = 0.35;
```

One way of plotting the function:

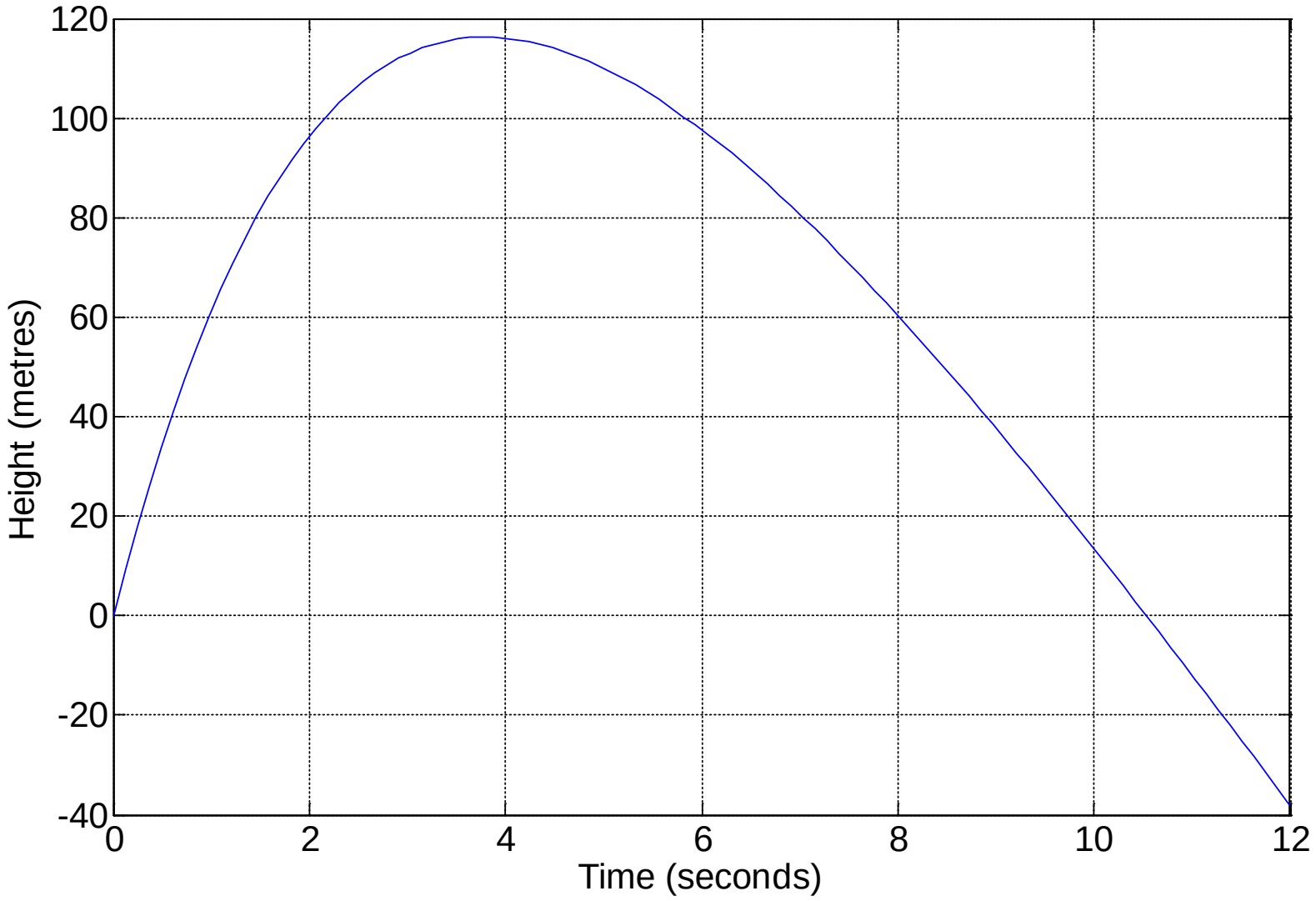
```
>> hFunc = @(t) objectHeight (v0, r, t);
>> fplot (hFunc, [0 12]);
```

Another way (works because the function can deal with a vector of times):

```
>> t = linspace(0,12,100);
>> h = objectHeight (v0, r, t);
>> plot (t, h);
```

Yet another way (doesn't require function to handle vectors):

```
>> t = linspace(0,12,100);
>> h = ones (size(t));    % preallocate for efficiency
>> for k = 1 : length(t)
       h(k) = objectHeight(v0, r, t(k));
   end
>> plot (t,h)
```


120
100
80
60
40
20
0
-20
-40
Height (metres)
0
2
4
6
8
10
12
Time (seconds)

Part (i) solution:

```
>> f = @(t) objectHeight(v0, r, t) - 80;
>> time1 = fzero (f, [0.1, 2])
time1 =
    1.4520
>> time2 = fzero (f, [6, 8])
time2 =
    7.0315
```

Part (ii) solution:

```
>> f = @(t) objectHeight(v0, r, t);
>> time3 = fzero (f, [10, 12])
time3 =
   10.5378
```

Part (iii) requires new techniques.

One possibly is to differentiate the height equation and set the result equal to zero.

$$\frac{dh}{dt} = \frac{1}{r}\left(v_0 + \frac{g}{r}\right)\left(re^{-rt}\right) - \frac{g}{r}$$

This reduces the problem to a root finding exercise:

```
>> g = 9.81;  % previously only defined in function
>> der = @(t) (1 / r) * (v0 + g/r) * (r * exp(-r * t)) - g / r;
>> tmax = fzero (der, [3 5])
t max=
   3.8014
>> maxHeight = objectHeight (v0, r, tmax)
maxHeight =
  116.3098
```

Note: In general a point at which the derivative is zero can be either a minimum, a maximum, or an inflection point. In this case our plot makes the situation clear.

Another possibility is to use function *fminbnd* as shown below:

```
>> f = @(t) -objectHeight (v0, r, t);    % note negative sign
>> tmax = fminbnd (f, 3, 5)
tmax =
   3.8014
>> maxHeight = objectHeight (v0, r, tmax)
maxHeight =
  116.3098
```

Function *fminbnd*:

```
minX = fminbnd (f, lowX, highX);
```

f = a function of one variable
lowX = lower limit of region containing minimum
highX = upper limit or region containing minimum

minX = location of minimum

Note: Maximums can be found by negating the function of interest (as in this example).

Another way to use function *fminbnd*, with two outputs

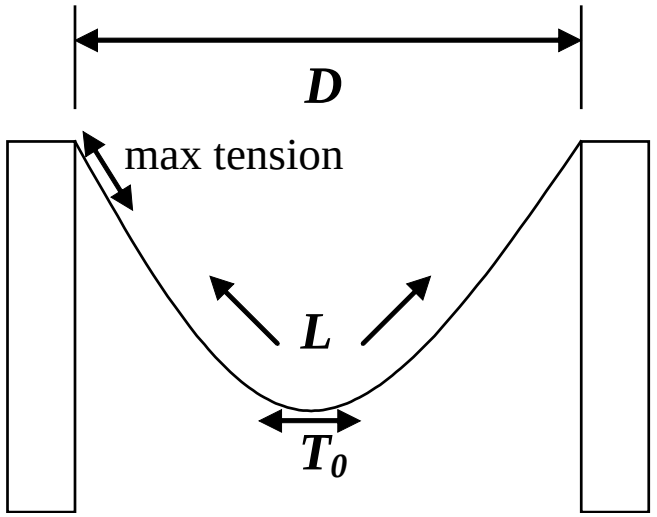
```
>> f = @(t) -objectHeight (v0, r, t);    % note negative sign
>> [tmax, maxHeight] = fminbnd (f, 3, 5)
tmax =

    3.8014


maxHeight =

 -116.3098
```

Problem: A cable weighing *m* kg/m is to be hung between two points *D* metres apart. What cable length *L* will give the least maximum tension ?

Assume *D* = 100m and *m* = 3 kg/m.

$$L = \frac{2T_0}{mg}\sinh\left(\frac{mg}{T_0}\frac{D}{2}\right)$$

$$T_{\mathrm{MAX}} = T_0 \cosh\left(\frac{mgD}{2T_0}\right)$$



where *L* is the length of the cable (in metres)
T_0 is the tension at the lowest point in the cable (in Newtons)
T_{MAXC} is the maximum tension in the cable (in Newtons)
m is the mass per unit length of the cable (in kg/m)
g is the acceleration due to gravity (9.81 m/sec^2)
D is the distance between the two points (in metres)

```
function [ Tmax ] = maxTension( L, m, D )
%calcT0 Calculates maximum cable tension
% Inputs: L = cabla length (in m)
%         m = mass per unit length (in kg/m)
%         D = distance between supports (in m)
% OutputsL Tmax = maximum tension (in N)

g = 9.81;
calcL = @(T0) ((2 * T0) / (m * g)) * sinh( m * g * D / (2 * T0));

lo = eps; hi = 1; % note calcL does not work for T0 = 0
while calcL(hi) > L  % tension not high enough
   lo = hi; hi = 2 * hi;
end

f = @(T0) calcL(T0) - L;

T0 = fzero(f, [lo hi]);

Tmax = T0 * cosh( m * g * D / (2 * T0));

end
```

```
D = input ('Enter D: '); m = input ('Enter m: ');

Tfunc = @(L) maxTension (L, m, D);

fplot (Tfunc, [D * 1.01, D * 2]);
grid on;
xlabel ('Length (metres)');
ylabel ('Max tension (Newtons)');

% bracket the minimum
L1 = D * 1.001; L2 = D * 1.002; T2 = Tfunc(L2);
while 1
   L3 = L2 + 1; T3 = Tfunc(L3);
   if T3 > T2  % tension has started going up
      break;
   end;
   L1 = L2; L2 = L3; T2 = T3;
end

bestL = fminbnd (Tfunc, L1, L3);

fprintf ('Best length is %4.2fm.\n', bestL);
fprintf ('Max tension is %4.2fN.\n', Tfunc(bestL));
```

Key points to note:

The “function” in a root finding or optimization problem need not correspond to a simple mathematical expression.

Instead it can be any process which converts an input value into an output value.

A function can involve hundreds of lines of code and complex procedures.

Function evaluation can take a significant amount of time (hours, in extreme cases).

It is desirable to keep the number of function evaluations required by a procedure to a minimum.

Save function results that will be required later instead of re-evaluating the function . Example: Compare the bisection search in these notes (bisect.m, one function evaluation per iteration) with the one in the Chapra text (p 127, two function evaluations per iteration).

When function evaluations take a lot of time faster search methods become relatively more attractive.