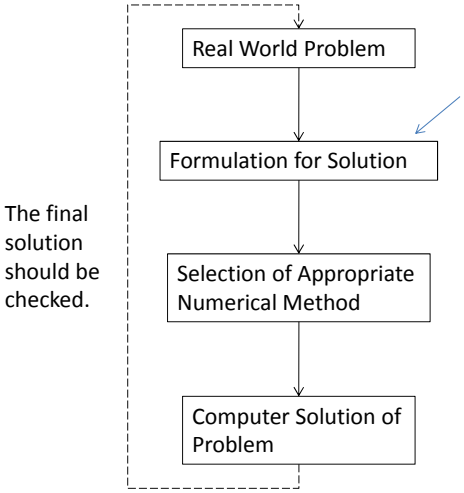
ECOR 2606 – The Big Picture



Apply engineering knowledge to get desired output in terms of known inputs.

The final solution should be checked.

Problem: A skydiver jumps from a balloon (hence no initial velocity).

Downward force due to gravity: $F_D = mg$

Upward force due to air resistance: $F_U = c_d v^2$

What is the skydiver's velocity as a function of time?

A little modelling produces a differential equation:

$$a = \frac{F}{m} = \frac{F_D - F_U}{m} = \frac{mg - c_d v^2}{m}$$

$$\frac{dv}{dt} = g - \frac{c_d}{m} v^2$$

An analytical solution can be obtained fairly easily:

$$v(t) = \sqrt{\frac{gm}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}} t\right)$$

But suppose that the math was a little too scary….

In this case we could resort to using numerical methods.

The following becomes exact as Δt approaches zero.

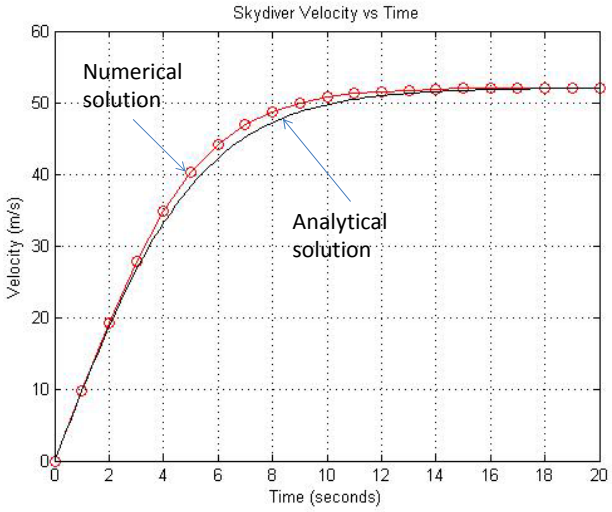
$$\frac{dv}{dt} = \frac{\Delta v}{\Delta t} = \frac{v_{i+1} - v_i}{t_{i+1} - t_i}$$

For small Δt it's not exact but if Δt is made small enough it's a good approximation. Rearranging and substituting for dv/dt gives

$$v_{i+1} = v_i + \frac{dv}{dt}\Delta t = v_i + \left(g - \frac{c_d}{m} v_i^{\,2} \right) \Delta t$$

Starting from the known initial conditions ($t_1 = 0$, $v_1 = 0$), this formula and some assumed Δt can be used to calculate v_2, v_3, v_4 and so on.

This is an example of *Euler's Method*.



Numerical solution obtained using Euler's Method and 1 sec intervals.

The Matlab code used to produce the graph. (EulerVelocity.m)

```
tMax = 20; % maximum time

m = 69.1; g = 9.81; cD = 0.25;

% Euler solution
n = 21; % number of points (including initial point)
deltaT = tMax / (n - 1); % interval width
t = linspace (0, tMax, n);
v = zeros (size(t));
for i = 2:n
   v(i) = v(i - 1) + (g - ((cD / m) * v(i - 1)^2)) * deltaT;
end

% analytical solution
tFine = linspace (0, tMax, 100);
vCalc = sqrt(g * m / cD) * tanh(sqrt(g * cD / m) * tFine);

plot (t, v, 'r-o', tFine, vCalc, 'k');
title ('Skydiver Velocity vs Time');
xlabel ('Time (seconds)'); ylabel ('Velocity (m/s)');
grid on
```

The skydiver's velocity asymptotically approaches a *terminal velocity.*

At this velocity the downward and upward forces are equal and there is therefore no net force and no acceleration.

Downward force = $F_D = mg$ = Upward force = $F_U = c_d v^2$

Setting $mg = c_d v^2$ and solving for v gives $v(t) = \sqrt{\frac{gm}{c_d}}$

The same result can be obtained by noting that in the steady state $dv/dt = 0$

$$\frac{dv}{dt} = 0 = g - \frac{c_d}{m} v^2$$

This kind of analysis can provide a very useful check on the validity of a solution (i.e. if a solution doesn't give the correct result as *t* approaches infinity there's something wrong with it).

In our case both the analytical and Euler solutions approach the correct value.

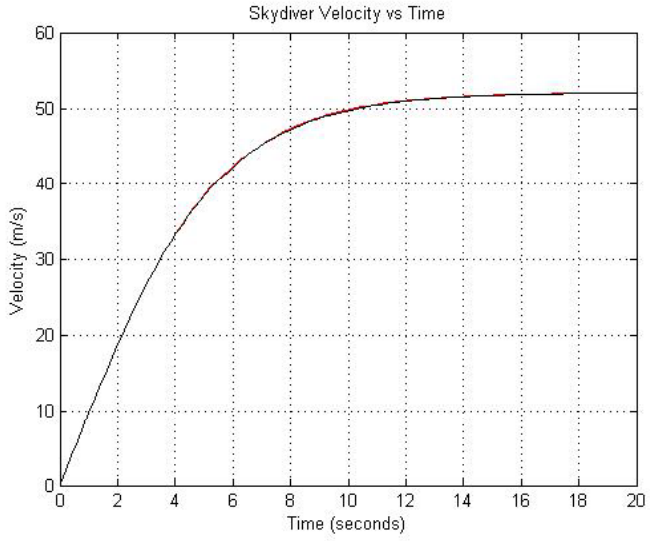
Euler's Method is very much a "low end" technique for solving differential equations.

There are much more sophisticated and accurate techniques.

One second intervals are also pretty coarse.

With smaller intervals even Euler's method would produce much better results.

The following slide shows the results of using 0.2 second intervals and a second/third order Runge-Kutta technique.



Analytical solution in black, numerical solution (using ode23) in red.

The Matlab code used to produce the graph. (ODEVelocity.m)

```
tMax = 20; % maximum time

m = 69.1; g = 9.81; cD = 0.25;

% analytical solution
tFine = linspace (0, tMax, 101);
vCalc = sqrt(g * m / cD) * tanh(sqrt(g * cD / m) * tFine);

% ODE Solver solution
dvdt = @(t,v) g - ((cD / m) * v^2);
[tt vv] = ode23(dvdt, linspace (0, tMax, 101), 0);

plot (tt, vv, 'r', tFine, vCalc, 'k');
title ('Skydiver Velocity vs Time');
xlabel ('Time (seconds)'); ylabel ('Velocity (m/s)');
grid on
```

Note that although the results are much better the code is actually shorter. This is because instead of doing the work ourselves we are making use of a Matlab function (ode23) that does everything for us.

Problem: How far will the skydiver fall is some given time?

The distance travelled by the skydiver can be obtained by integrating velocity.

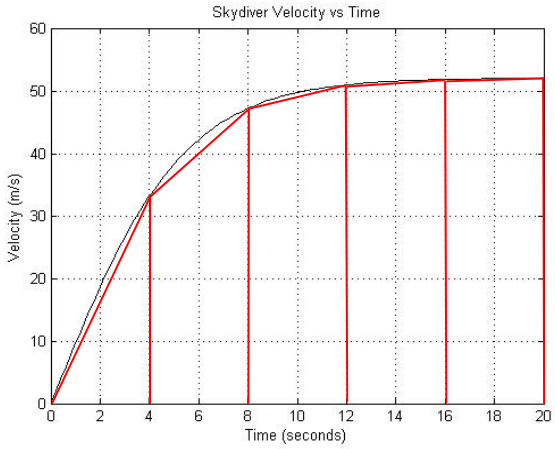
$$d(T) = \int_0^T v(t)dt = \int_0^T \sqrt{\frac{gm}{c_d}} \tanh h\left(\sqrt{\frac{gc_d}{m}}t\right)dt$$

This integral can be calculated analytically. The answer is

$$d(T) = \frac{m}{c_d}\ln\left(\cosh\left(\sqrt{\frac{gc_d}{m}}\ T\right)\right)$$

If the integral was too complex to deal with, numerical methods could be employed.

One possibility is trapezoidal integration.



Area under the velocity curve = value of the integral = distance fallen.

Area under curve ≈ the total area of the five trapezoids outlined in red.

Accuracy increases as the number of trapezoids in increased.

Distance at t = 20 seconds

Analytical = 849.999823

Intervals	Numerical
1	520.163630
5	836.661239
9	845.946854
13	848.063507
17	848.868869
21	849.259103
25	849.477339
29	849.611609
33	849.700056
37	849.761388
41	849.805655
45	849.838648
49	849.863893
53	849.883640
57	849.899376
61	849.912119

The Matlab code used to produce the results . (TrapDistance.m)

```
T = 20; % time of interest

m = 69.1; g = 9.81; cD = 0.25;

% analytical solution
distanceA = (m / cD) * log(cosh(sqrt((g * cD)/m) * T));
fprintf ('Analytical = %f\n', distanceA);

% numerical solution
fprintf ('Intervals    Numerical\n');
for n = 2: 4: 62; % n = number of points
  t = linspace (0, T, n);
  v = sqrt(g * m / cD) * tanh(sqrt(g * cD / m) * t);
  distanceN = trapz(t, v);
  fprintf ('%5d       %10.6f\n', n - 1, distanceN);
end
```

Note: “trapz” is a Matlab function that does trapezoidal integration.

Suppose that the skydiver jumps from 1000m above the ground and we would like to know what the time to impact will be if his parachute fails to deploy.

We have a formula that converts time to distance:

$$d(T) = \frac{m}{c_d} \ln\left(\cosh\left(\sqrt{\frac{gc_d}{m}}\, T \right) \right)$$

We need a formula that converts distance to time:

$$t(D) = ???$$

In this case the original formula can fairly easily be “inverted”:

$$t(D) = \cosh^{-1}\left(e^{\frac{Dc_d}{m}} \right) \Big/ \sqrt{\frac{gc_d}{m}}$$

In cases where this is not possible, numerical methods can be employed.

We want to find time *T* such that:

$$d(T) = \frac{m}{c_d} \ln\left(\cosh\left(\sqrt{\frac{g c_d}{m}}\, T \right) \right) = 1000 \text{ metres}$$

Or, to put it a slightly different way, we want to find the *T* that satisfies:

$$\frac{m}{c_d} \ln\left(\cosh\left(\sqrt{\frac{g c_d}{m}}\, T \right) \right) - 1000 = 0$$

We now have a "root finding" problem.

It can be solved numerically using iterative techniques that keep "guessing" at *T* (using an enlightened strategy) until a satisfactory answer is obtained.

The enlightened strategy reduces the number of guesses required to close in on the answer and in any case computers can do a lot of calculations very quickly.

In Matlab we need only the following commands:

```
f = @(T) (m / cD) * log(cosh(sqrt((g * cD)/m) * T)) - height;
time = fzero (f, [0, 600]);
fprintf ('Root Finding: The skydiver will impact in %f seconds.\n', time);
```

Function *fzero* uses numerical methods to locate roots.

The answer obtained is identical (to six decimal places) to that given by the analytical formula.

```
Root Finding: The skydiver will impact in 22.882511 seconds.
Analytical: The skydiver will impact in 22.882511 seconds.
```

See Matlab file ImpactTime.m