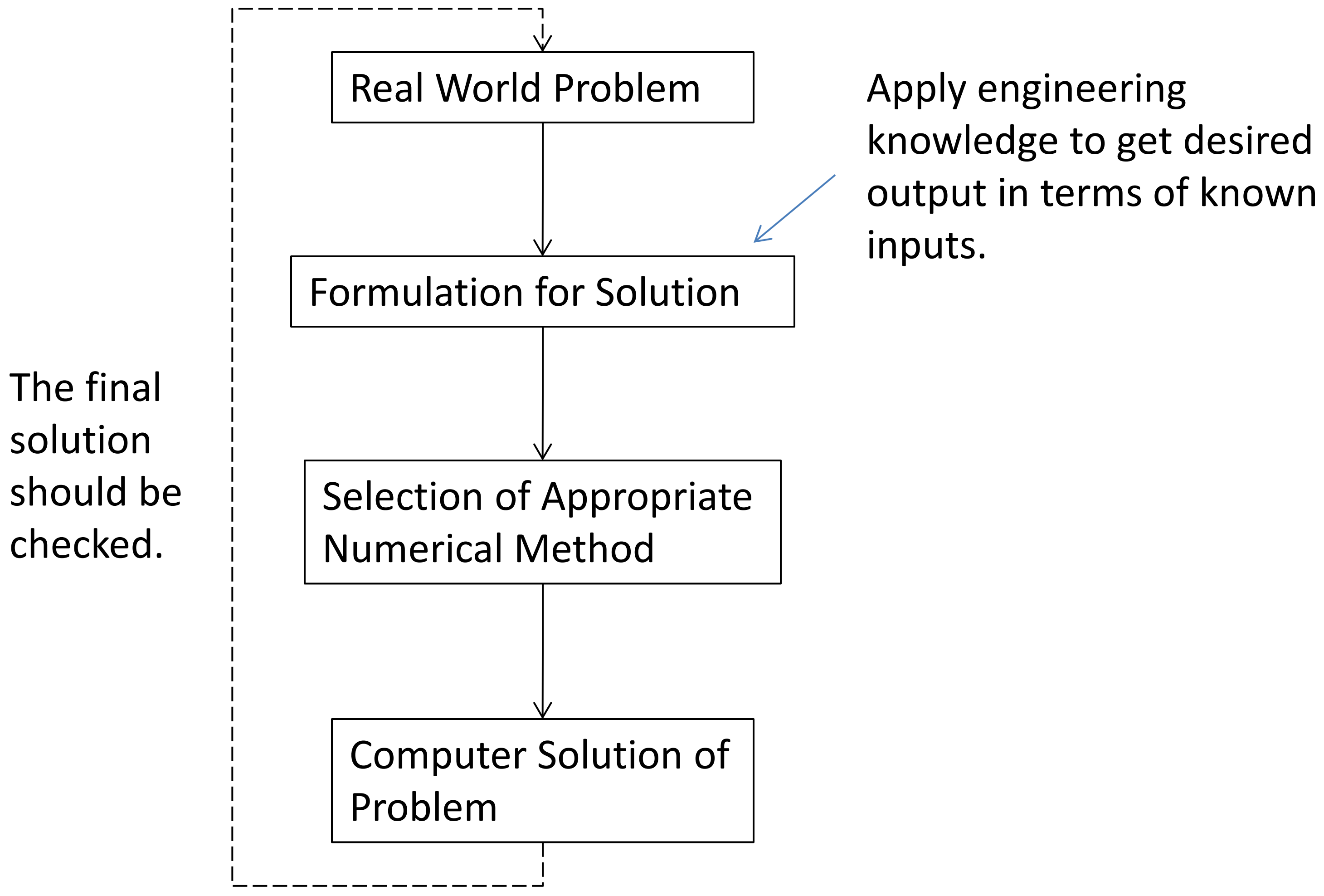
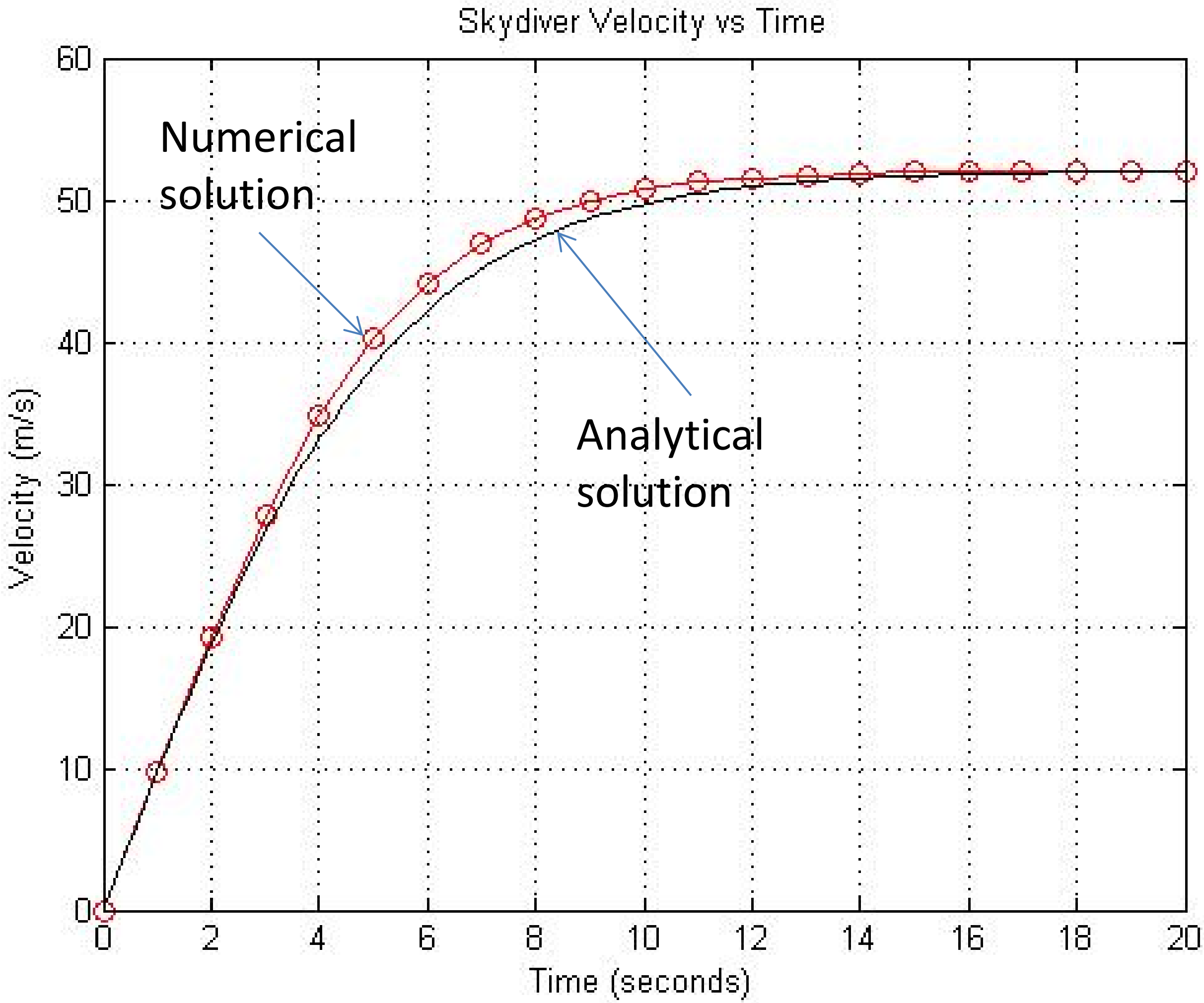
ECOR 2606 – The Big Picture





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

The Matlab code used to produce the graph .

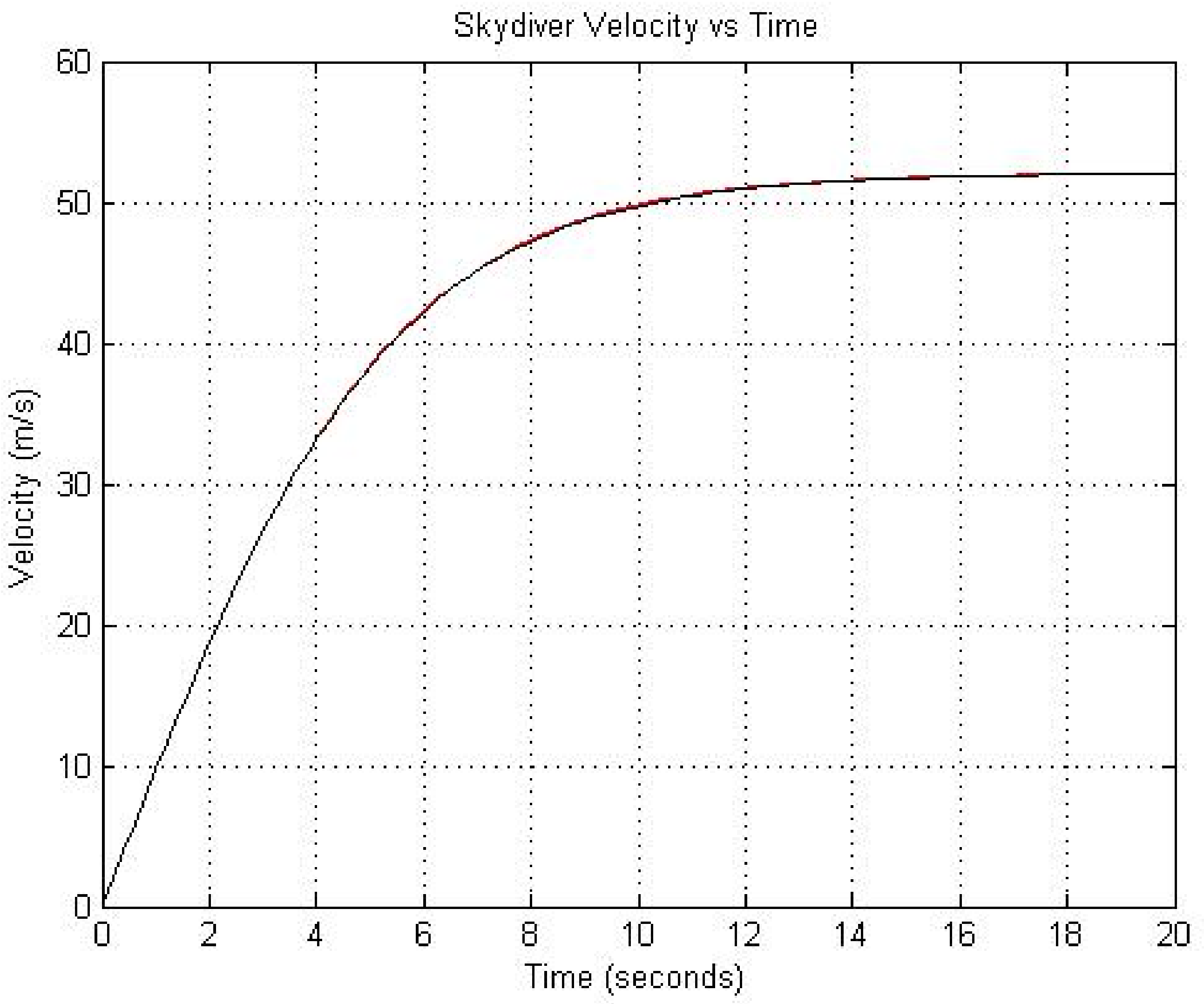
```
tMax = 20; % maximum time

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
```



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

The Matlab code used to produce the graph .

```
tMax = 20; % maximum time

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

% analytical solution
tFine = linspace (0, tMax, 100);
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, 100), 0);

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

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

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

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

See text p 393.

Distance at t = 10 seconds

Analytical = 335.445801

Intervals	Numerical
1	248.601887
11	334.829059
21	335.276734
31	335.368231
41	335.401458
51	335.417144
61	335.425770
71	335.431015
81	335.434441
91	335.436800
101	335.438494
111	335.439751
121	335.440710
131	335.441458
141	335.442052

The Matlab code used to produce the results .

```
T = 10; % 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: 10: 142; % 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
```