# Non-Constant Acceleration

Kaylin Shanahan 2023

---

This is a Python notebook to estimate the velocity and displacement of a car over a period of 10 seconds. This will be done using Euler's technique, and display the results on graphs of the numerical and exact analytical solutions, and also print the value of displacement.

The car begins its acceleration from rest, but this acceleration is not constant. The acceleration of this car can be described by the following formula:

$$\frac{d^2x}{dt^2} = a_{max}(1 - e^{-\frac{t}{\tau}})$$

Where $a_{max} = 4.0ms^{-2}$ and $\tau = 5.0s$

The Euler programme in this case will be used to describe the situation with a car where the acceleration changes. $X$ will be used to measure position (displacement), in relation to $t$, time.

By integrating the expression for acceleration once, we get an equation for velocity $(v)$ :

$$\frac{dx}{dt} = xa_{max}(1 - e^{-\frac{t}{\tau}})$$

Integrating this again, gives an expression for displacement $(x)$ :

$$x(t) = \frac{1}{2}x^2a_{max}(1 - e^{-\frac{t}{\tau}})$$

Using these expressions, it is possible to find analytical solutions for velocity and displacement.

- Input the initial conditions

- Determine the number of steps required
- Initialise x and t values
- Calculate the velocity and displacement using the differential equation above, both numerically and analytically
- Output the x and t values
- Output a plot of the numerical and analytical solutions on a graph

In [91]:

```
# Solve for velocity and displacement of car with non-constant acceleration using Euler's technique
# The 2nd order ODE is: d^2x/dt^2 = a_max (1 - np.exp(- t/tau))

# Numpy is needed for the natuaral log and exponential function
import numpy as np
# Matplotlib is needed to plot graphs
import matplotlib.pyplot as plt

# Initial conditions
a_max = 4.0              # Maximum acceleration in meters per second squared
tau = 5.0                # Time constant in seconds

# Time values required for programme
t_max = 10               # Iterate up to a maximum time of 10 seconds
delta_t = 0.1            # Set the time step in seconds
N = int(t_max/delta_t)   # Calculate the number of time jumps in minutes

#Set up NumPy arrays to hold the displacemtent x, velocity v and time t values
# We need one element more in each array, than the number of jumps
x = np.zeros(N + 1)
v = np.zeros(N + 1)
t = np.zeros(N + 1)

# Initialise the zeroth values
x[0] = 0                 # Initial displacement
v[0] = 0                 # Initial velocity
t[0] = 0                 # Initial time
```

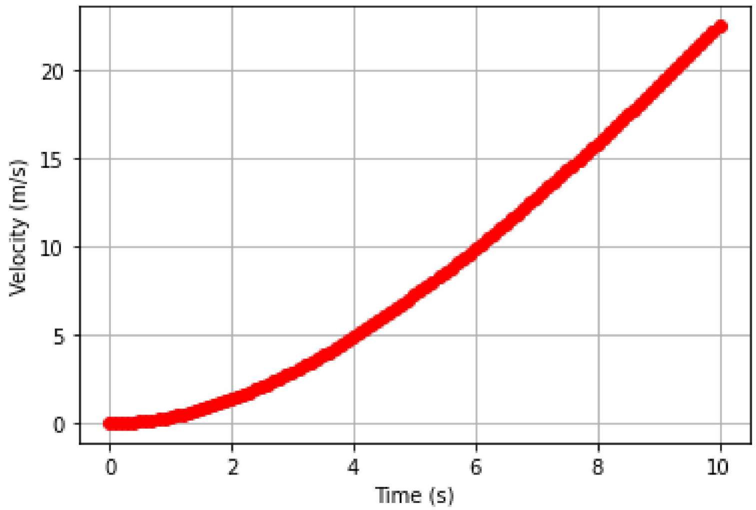
In [92]:

```
# Use a for loop to step along the t-axis
for i in range(N):
    a = a_max * (1 - np.exp(-(t[i]/tau))) # Use the ODE to calculate the acceleration
    v[i+1] = v[i] + a * delta_t           # Estimate v at the end of the interval
    x[i+1] = x[i] + v[i] * delta_t        # Estimate x at the end of the interval
    t[i+1] = t[i] + delta_t               # Calculate t at the end of the interval
```

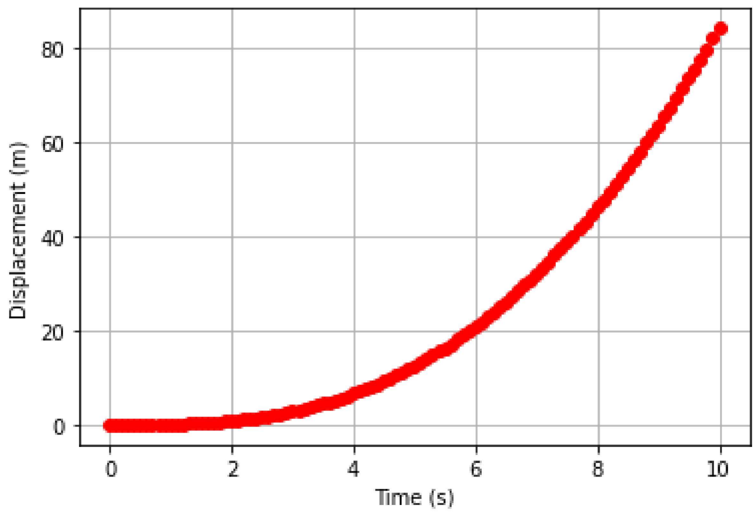
In [93]:
```
print("Displacement after 10 seconds = {0:8.2f} m".format(x[N]))
print("Velocity after {0} seconds from Euler = {1:8.2f} m s^-1".format(t_max, v[N]))
```

```
Displacement after 10 seconds =    84.20 m
Velocity after 10 seconds from Euler =    22.53 m s^-1
```

In [94]:
```
# Numerical solution for velocity
# Plot the numerical solution as points
plt.plot(t, v, "ro")
plt.xlabel("Time (s)")
plt.ylabel("Velocity (m/s)")
plt.grid()
```

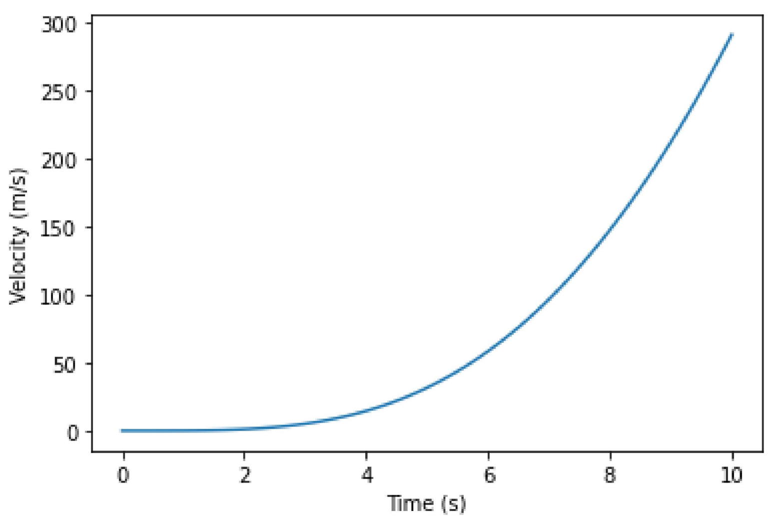



In [95]:
```
# Numerical solution for displacement
# Plot the numerical solution as points
plt.plot(t, x, "ro")
plt.xlabel("Time (s)")
plt.ylabel("Displacement (m)")
plt.grid()
```




In [96]:

```
# Analytical solution for velocity
# Plot the analytical solution as a line
# Use the same (N+1) t values
v_analytic = x * a_max * (1 - np.exp(- (t / tau)))
plt.xlabel("Time (s)")
plt.ylabel("Velocity (m/s)")
plt.plot(t, v_analytic)
plt.show()
```




In [98]:

```
# Analytical solution for displacement
# Plot the analytical solution as a line
```

```
# Use the same (N+1) t values
x_analytic = ((x/2)**2) * a_max * (1 - np.exp(- (t / tau)))
plt.xlabel("Time (s)")
plt.ylabel("Displacement (m)")
plt.plot(t, x_analytic)
plt.show()
```