Objectives

This tutorial has two objectives:

1. To have the students build an function that numerically solves 2nd order ordinary differential equation (ODE) based on the *midpoint* algorithm.
2. Use their ODE solver to make a preliminary investigation of oscillating systems.

Anonymous Functions

MATLAB give the user the ability to create their own functions that behave in the same way as MATLAB's built in functions like `sin()` and `exp()`. The simplest way to do this is using an anonymous function. If we wanted to make an anonymous function for $y(t)$ that respresents a freely falling object, you're code might look like:

```
g = 9.8; % m/s^2
y0 = 5;  % m
v0 = 2;  % m/s

y = @(t) y0 + v0*t + 0.5*g*t.^2;
```

Now, you can use `y` like a function:

```
>> y(3)

ans =

   55.1000
```

If you've coded the anonymous function properly (notice the `.^`), then you can even input arrays:

```
>> t = linspace(0,10,1000);
>> y(t)

ans =

    5.0000   13.2716   33.6420   66.1111  110.6790  167.3457  236.1111 . . .
```

Important: `y` is **not** an array. It has been defined as a function, so when plotting it you'll need to use code that looks like:

```
plot(t,y(t));
xlabel('t', 'fontsize',20);
ylabel('y(t)', 'fontsize',20);
```

Now perform the following tasks:

1. Make an anonymous function for a quadratic drag ($|\vec{F}_d| = \alpha v^2$) model. Recall that the terminal velocity occurs when the drag force balances gravity to yield zero net force.

$$v_T = \sqrt{\frac{mg}{\alpha}}$$

In terms of g and v_T, the position as function of time is

$$y(t) = y_0 + \frac{v_T^2}{g}\left[\log\left(\cosh\left(\frac{gt}{v_T} + \tanh^{-1}\left(\frac{v_0}{v_T}\right)\right)\right) + \frac{1}{2}\log\left(1 + \frac{v_0^2}{v_T^2}\right)\right] \tag{0.1}$$

where $y_0 = 5$ m/s, $v_0 = 2$ m, $\alpha = 0.3$ Ns^2/m^2, $m = 1$ kg, and $g = 9.8$ m/s^2. Plot this function and label the axes. Overlay a plot of the same scenario with $\alpha = 0$, i.e. no air drag.

2. Next, define an anonymous function of two input variables t and v_T the represents the object's velocity as a function of t and the parameter v_T.

$$v(t, v_T) = v_T \tanh\left(\frac{gt}{v_T} + \tanh^{-1}\left(\frac{v_0}{v_T}\right)\right) \tag{0.2}$$

Make a series of plots of v versus t for a fixed v_T, one for of 10 values of v_T that lie in the range 1 m/s to 100 m/s. Overlay each of plots on the same figure. (hint: Use a `for` loop.)

Differential Equation Solver

Building `evolve()`

Write a MATLAB function called `evolve()` that is based on the midpoint algorithm for numerically solving ordinary differential equations. I should accept three arguments:

- An array `t` with size $1 \times N$ that represents t.
- The initial position `x0` (not an array).
- The initial position `v0` (not an array).
- The name of the anonymous function that computes the acceleration. This function must have three arguments, that is it must have the form $f_a(t, x, v)$.

and return three arrays `x`, `v`, and `a` that represent position, velocity and acceleration. Each of these arrays must be the same size as the time array `t`. The header for your function should look like:

```
function[x, v, a] = evolve(t, x0, v0, f_a)

        insert your code here ....

end
```

The midpoint algorithm performs a time step in two parts. First, we evolve to a time half way between time values using a Euler time step,

$$\begin{aligned} t_{\frac{1}{2}} &= t_{i-1} + \frac{dt}{2} \\ v_{\frac{1}{2}} &= v_{i-1} + a_{i-1}\frac{dt}{2} \\ x_{\frac{1}{2}} &= x_{i-1} + v_{i-1}\frac{dt}{2} \end{aligned} \tag{0.3}$$

and the use these to find $a_{\frac{1}{2}}$

$$a_{\frac{1}{2}} = f_a(t_{\frac{1}{2}}, x_{\frac{1}{2}}, v_{\frac{1}{2}})\,.$$

Next, we use the halfway values to go the who time step

$$\begin{aligned} t_i &= t_{i-1} + dt \\ v_i &= v_{i-1} + a_{\frac{1}{2}}dt \\ x_i &= x_{i-1} + v_{\frac{1}{2}}dt \end{aligned} \tag{0.4}$$

and then find a_i:

$$a_i = f_a(t_i, x_i, v_i)\,.$$

We repeat for each element of the time array using a `for` loop.

Testing the `evolve()` function

Now that you've build an ODE solver, it's time to test it. We will test it by choosing a physical system that has a known analytic solution, and then comparing your numerical solution with the analytic one. We will choose a ball dropped from the top of the Empire State building as our physical system. Take the initial conditions to be:

$y_0 = 443$ m (height of the Empire State building)
$v_0 = 0$ m/s

1. Write an anonymous function for the acceleration of this system, evolve it using your ODE solver, and then plot $y(t)$. Overlay the analytic version of $y(t)$ and test that our code works.

2. Repeat using a quadratic drag model with $|\vec{F}_d| = \alpha v^2$ with $\alpha = 0.3$ Ns^2/m^2. Use the same physical constants given in the previous section and the analytic formula is given above in eq. (0.1).

Oscillations

We will now use your `evolve()` function to model a shock absorber, and we will do this in steps of increasing complexity. A car's suspension usually consists of a shock absorber surrounded by a spring, as shown in figure 1 below, and together they form a damped oscillating system.

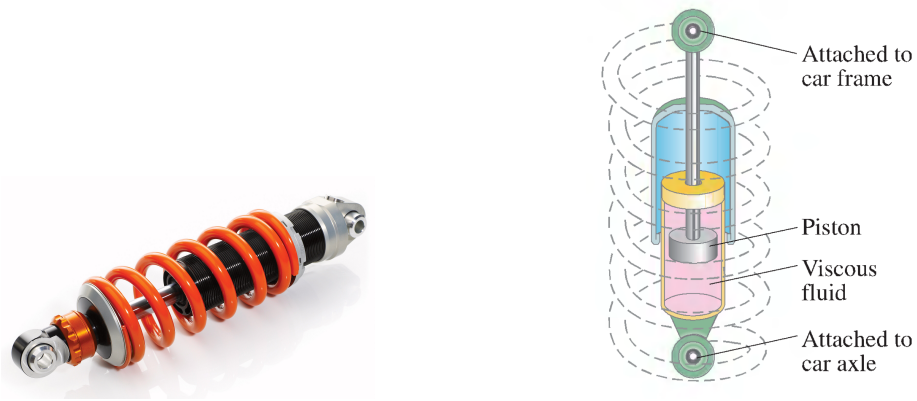

Figure 1: A picture and schematic of a spring and a shock absorber.

Suppose you've just hit a bump and the spring has been compressed by a small amount. Assuming reasonable initial conditions, perform the following tasks. For each task, plot y vs. t and and v vs. y, on separate figures, and label them appropriately.

1. Model the action of the spring assuming no friction, air drag or driving force. Assume $k = 16,000$ N/m.

2. The fluid inside the piston of the shock absorber acts to damp the system, and usually we'll assume the drag force is linear in velocity. Add the action effects of linear drag $|\vec{F}_d| = \alpha v$ and choose α so that the damped system has the best characteristics. You can do this by choosing an α, then repeatedly running your code and plotting $y(t)$ until the graph has the desired properties. Be sure to explain the criteria you used for selecting the best α. You may assume the car has a mass of 1000 kg.

3. Finally, add in the affects of a bumpy road. A simple model that describes this scenario assumes the road have a sinusoidal shape given by

$$Y(t) = A\sin(2\pi f t)$$

where f is the frequency that we encounter the bumps and A the size of a typical bump. Make reasonable estimates of A and f and then modify the acceleration function to account for the bumpy road. On your plot of $y(t)$, overlay a plot of $Y(t)$. Comment on the effectiveness of your suspension system.

Animations

Make an animation for the plot of v vs. y for each of the three scenarios in the previous section. You can do this with the following code:

```
figure;
hold on;
for I = 2:length(t)
    plot(y(I), v(I), 'bo ', 'markersize',5);
    axis([ 1.2*min(y), 1.2*max(y), 1.2*min(v), 1.2*max(v)]);
    pause(0.01);
end
```