Week #8 : Second Order Differential Equations

Goals:

- Express real world situations in terms of second order linear differential equations.
- Describe the difference between homogeneous and nonhomogeneous second order linear differential equations.
- Use MATLAB to solve linear and nonlinear second order differential equations, both homogeneous and nonhomogeneous.

Generating Numerical Solutions with MATLAB

Problem. Search for "ordinary differential equations" in MATLAB help.

Problem. What form of differential equation does MATLAB assume we have?

Note that differential equation solvers are a big component of MATLAB. If using MATLAB after this course, you may have to do some reading to identify the properties of the equation you have, and what the most appropriate solving tool is.

Problem. Which of the solvers is recommended as a "first try" solver?

ode45

The first solver to reach for in MATLAB is `ode45`. To run it, we need

- the DE function **in form** $\dfrac{dy}{dt} = f(t, y)$;

- the time span for the solution/simulation, $[t_0, t_{\text{end}}]$; and

- the initial condition $(y(t_0))$

Problem. What will `ode45` compute for us, given that information?

Temperature Model - Newton's Law of Heating and Cooling

The temperature of an object, y, changes at a rate proportional to the temperature difference between object, y, and its environment, T_{ext}.

Problem. Translate this law into a mathematical statement.

$$\frac{dy}{dt} = -k(y - T_{\text{ext}})$$

Solution Interpretation: a **solution** to a differential equation is the same as a **prediction**.

```
% Define temp DE in the form dy/dt = f(t, y)
% and set other constants
k = 0.7;  % /min

T_ext = 20; % external/environment temp

DE = @(t, y)   -k (y - T_ext);
```

Example - Step 2

```
% Solve based on initial condition
y0 = 100;  % initial temperature

tspan = [0, 30];  % interval for solution

[t, y] = ode45(DE, tspan_, y0);

plot(t, y);
```

To decipher and work with the output, it is critical that you understand what MATLAB provides.

- The final values of `t` and `y` are
- `t` starts at
- `t` ends at
- `y` starts at

Basic Process

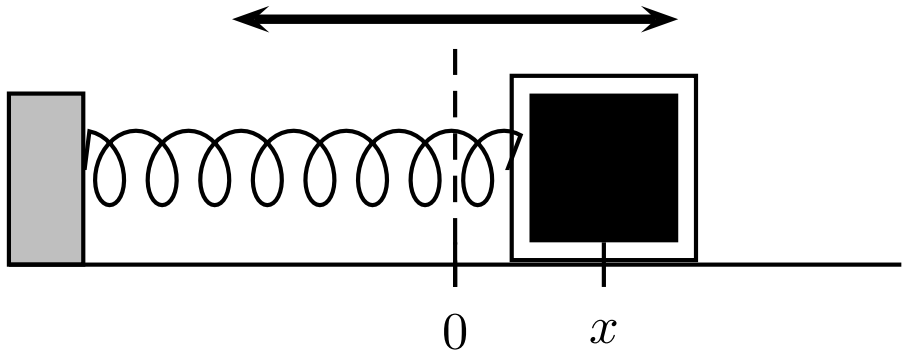
How does MATLAB do it?

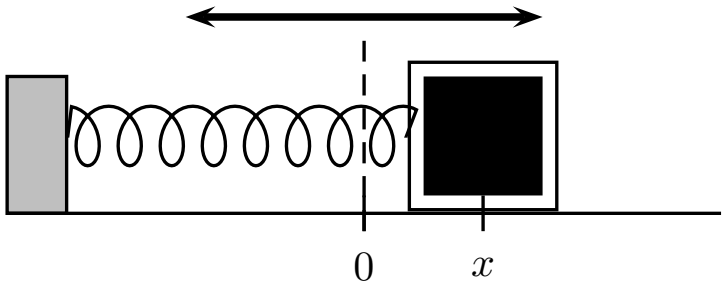
Second-Order Linear Equations - Spring System

So far we have seen examples of **first-order DEs**, or equations with first derivatives of some unknown function.

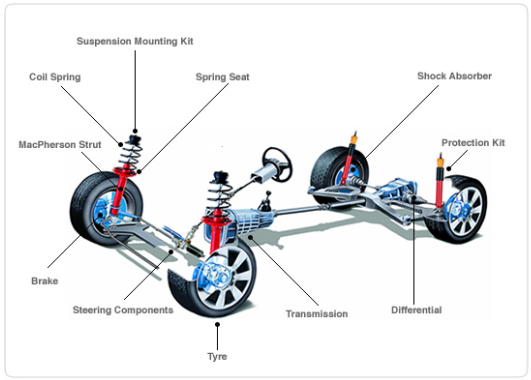
In the following examples, we will expand our study to differential equations with **second or higher derivatives**.

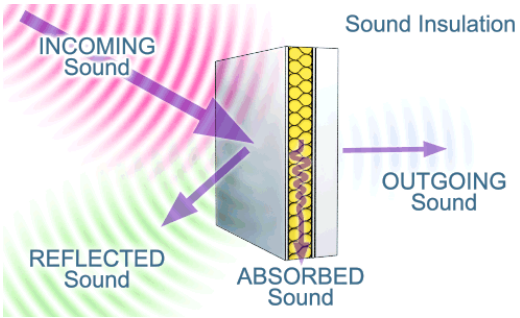
One classic source of differential equations of this type comes from analyzing the forces on a block at the end of a spring.

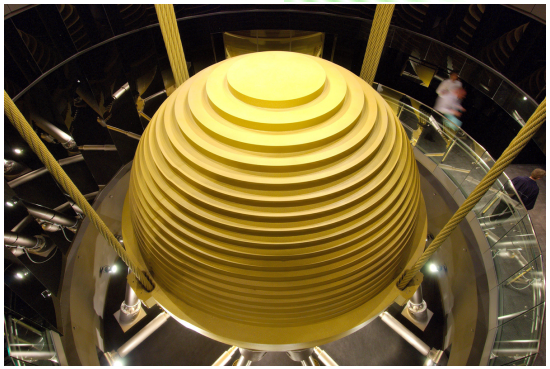




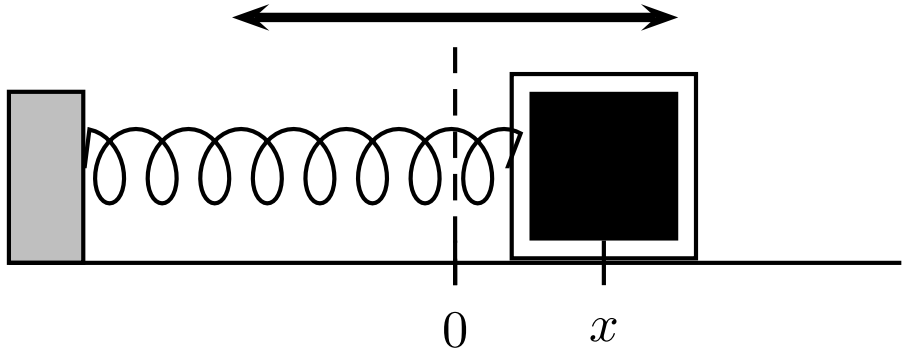
While the mathematics behind this simple system will be very interesting in their own right, we should also note at the outset that the simple spring/mass model can be applied to a wide variety of not-so-obviously related real-world problems.



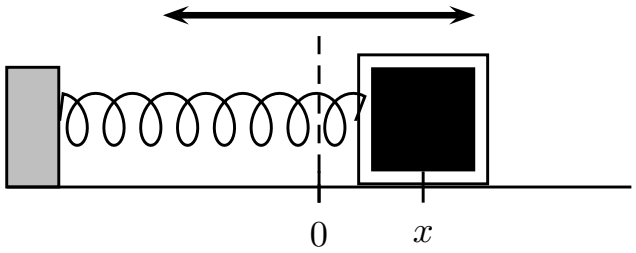




Spring System Analysis



Problem. In this system, how would you describe x in words?



Problem. Draw a free-body diagram for the mass. Indicate the magnitude of the forces, assuming

- the mass of the block is m kg, and
- the spring constant (in N/m) is given by the constant k.

Let us work with our intuition about this system before beginning the mathematics.

Problem. If the spring is very stiff, is k large or small?

Definition:: *Period* is the length of time to complete one full cycle/oscillation.

Problem. If we increase the stiffness of the spring, do you expect the *period* of the oscillations to increase or decrease? Why?

If we increase the mass, do you expect the *period* of the oscillations to increase or decrease? Why?

Problem. If we know k and m, and assume that damping is negligible, should we be able to determine the exact period of the oscillations?

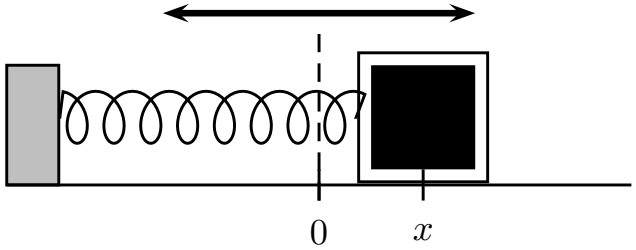
From the work so far, can we easily find the formula for the period?

The spring system is an excellent introduction to higher-order differential equations because

- we all have an intuition about how it *should* work physically,
- the mathematics and physics are simple, and
- there's no obvious way to predict critical features (e.g. the period) from the given information.

We clearly need some new tools!

Spring System as a DE



Problem. Use Newton's second law, $F = ma$, to construct an equation involving the position $x(t)$.

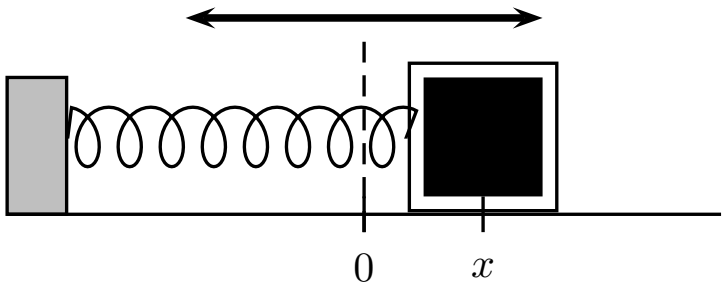
What order of differential equation does $F = ma$ produce for this spring/mass system?

To simplify matters temporarily, let us assume that both $k = 1$ N/m and $m = 1$ kg.

Problem. Rewrite the previous differential equation using those constants.

This differential equation invites us to find a function $x(t)$ whose second derivative is its own negative. What function(s) would satisfy that?

Problem. Having found two (and more) solutions to the differential equation for the spring/mass system, how does this family of solutions map back to the spring system?



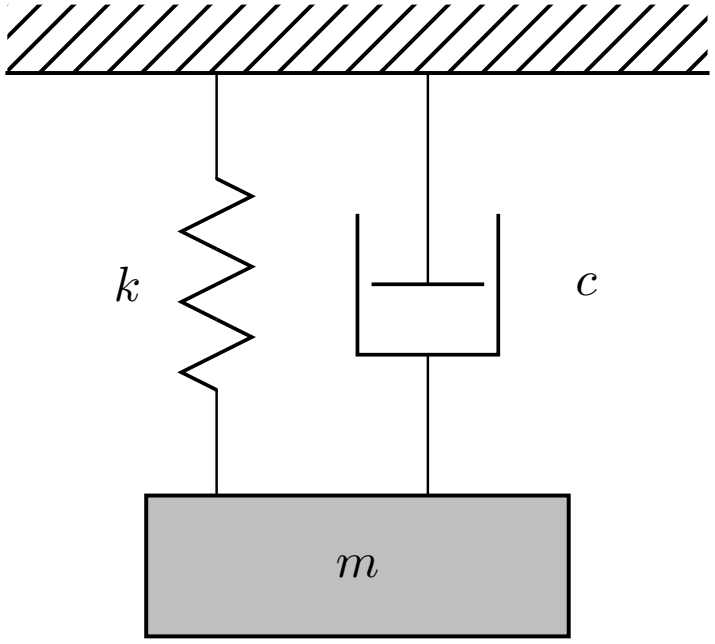
Is our solution an exact or a numerical solution?

Mechanical Vibrations - Spring-mass system

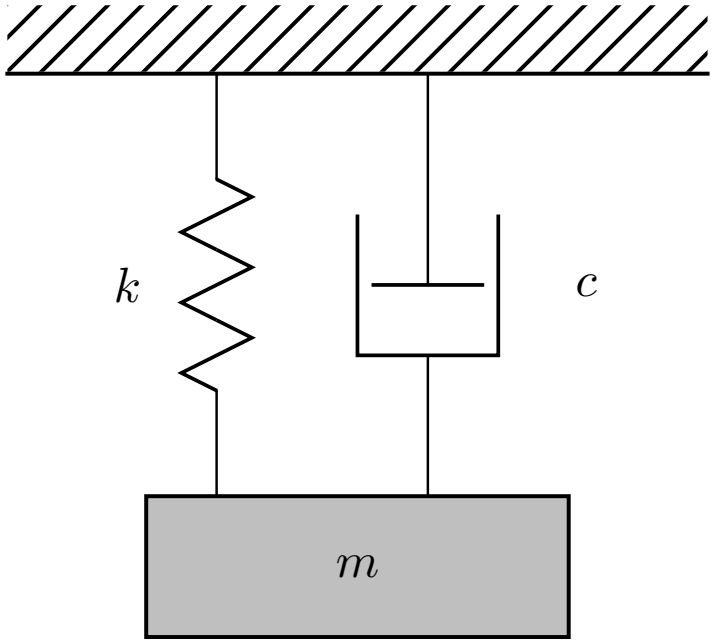
We now consider an extension to the spring/mass system.

Consider a mass m hanging on the end of a vertical spring; the spring has a natural length corresponding to $x = 0$.

We have now added a damper, which exerts a force proportional to the velocity of the oscillating mass.



Problem. Using Newton's second law, build a differential equation that governs the system.



Problem. Consider a system with a mass of 0.5 kg with spring constant $k = 2\ \mathrm{N} \cdot \mathrm{m}^{-1}$, with a damping coefficient of $c = 0.5$ N/(m/s). Assume the mass is displaced 0.4 m from equilibrium and released.

Problem. What differential equation dictates the motion of the mass?

Can we find an easy solution to this differential equation as we did with the undamped case?

Can MATLAB be used to build a numerical solution for the differential in its current form?

Converting Higher-Order DEs to 1st Order Systems

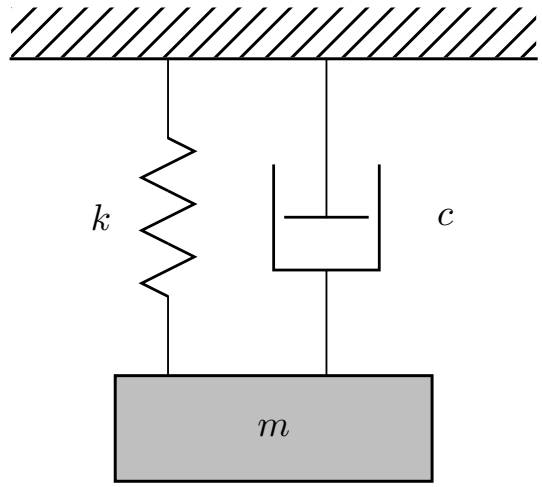
To solve a higher-order DE using MATLAB, we need to convert it first into a larger **first-order system** of differential equations.

Notation: In this section,

- vectors with be written vector hats, e.g. $\vec{w}, \vec{y}$,
- elements of a vector will be noted with subscripts, e.g. w_1, y_2, and
- other scalars will be in lower-case, e.g. c, λ.

Damped Spring DE:

$$mx'' = -cx' - kx$$



Problem. For the spring/mass DE, define a new vector of 2 variables, $\vec{w}$, that will allow the conversion of the second-order system to a first-order system.

Also see the MATLAB help menu entry on "Ordinary Differential Equations", referring to "Higher Order ODEs".

$$mx'' = -cx' - kx$$

Problem. Define the derivative of $\vec{w}$ in terms of $\vec{w}$ itself, making use of the DE as necessary.

This is now a **first-order system** of differential equations.
The variable we are most interested in for this example is $w_1 = x$, the *position* of the mass.

MATLAB Function Files

Our differential equation is now sufficiently complicated that writing it out in one line is difficult. We will need another MATLAB tool to proceed: a MATLAB function file.

Syntax A MATLAB function file

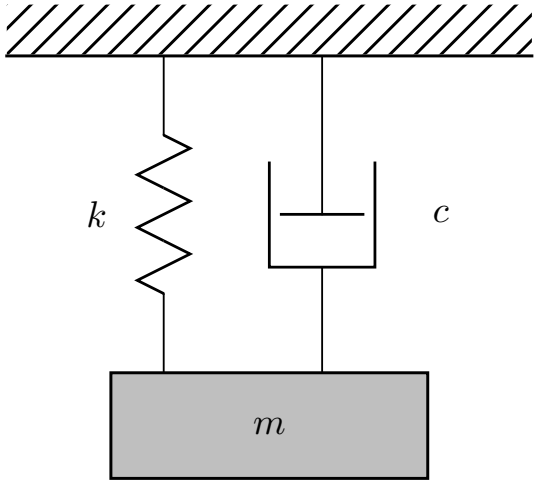
- has as its first line the form
  `function ret = f(....)`
- `ref` is the **return variable** and must be defined before the end of the function;
- `f` should be the same as the filename, with the `.m` extension;
- the `....` can be any list of input variables.

Problem. Define a new **MATLAB function `.m` file** called `hypotenuse` that takes in two lengths `a` and `b`, and returns the length of the hypotenuse for a right-angle triangle with side lengths a and b.

Problem. Define a new **MATLAB function `.m` file** called `springDE` which computes the derivative of $\dfrac{d\vec{w}}{dt}$.

$$\frac{dw_1}{dt} = w_2$$
$$\frac{dw_2}{dt} = -kw_1 - cw_2$$

Problem. Write a MATLAB script that graphs the solution (i.e. predicted motion) for the system described earlier:
A system with a mass of 0.5 kg with spring constant $k = 2\ \mathrm{N} \cdot \mathrm{m}^{-1}$, with a damping coefficient of $c = 0.5\ \mathrm{N/(m/s)}$. Assume the mass is displaced 0.4 m from equilibrium and released.
Include the graphical output from MATLAB on the next page.

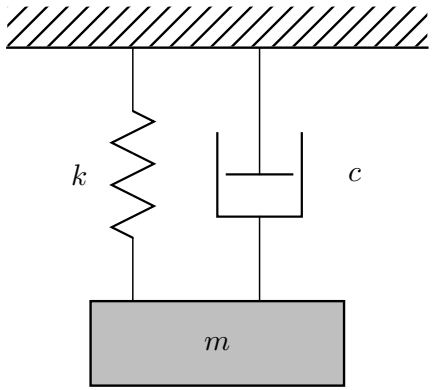

k
c
m

Unforced Spring/Mass System - Patterns of Behaviour

Using MATLAB, we can now easily study the impact of different masses, spring constants, or damping strength on the behaviour of a spring/mass system.

Damped Spring DE:

$$mx'' = -cx' - kx$$

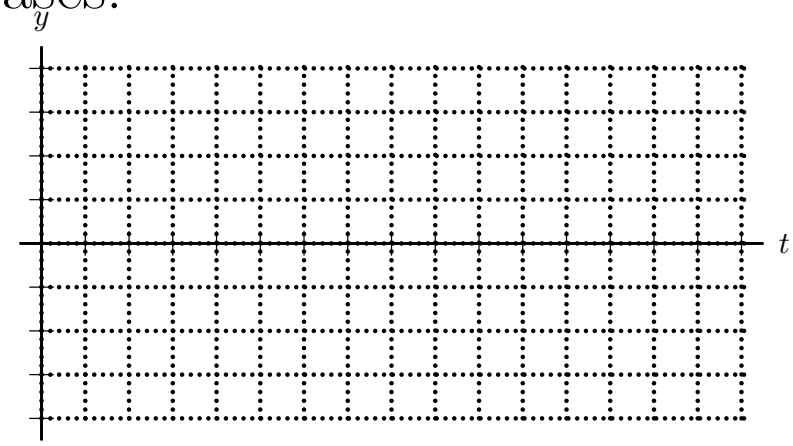


A famous result in physics is that when the damping coefficient:

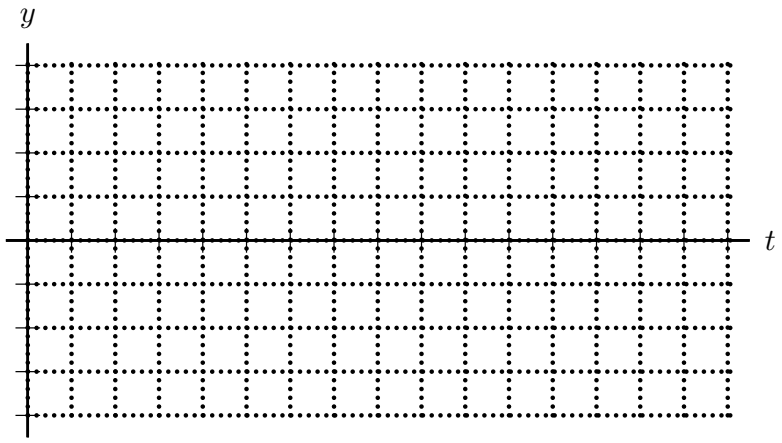
- c is 0, the system will be **undamped**;
- c is **less** than $\sqrt{4km}$, the system will be **under-damped**;
- c is **equal** to $\sqrt{4km}$, the system will be **critically damped**;
- c is **greater** than $\sqrt{4km}$, the system will be **over-damped**;

Problem. Using a spring constant of $k = 25$ N/m and mass $m = 1$ kg, find the critical damping level.

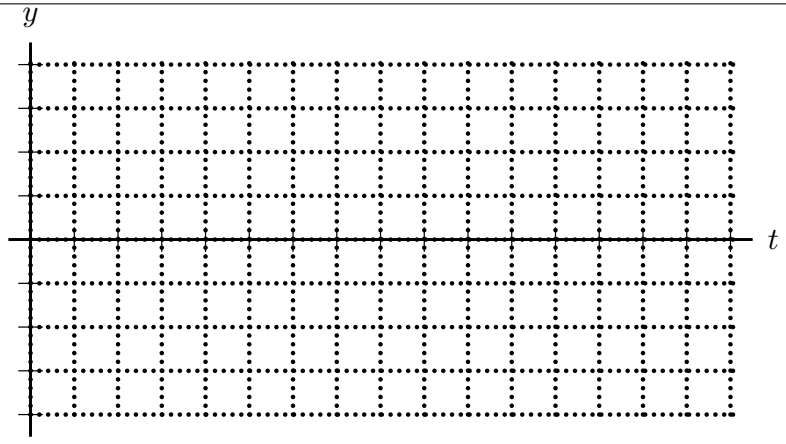
Problem. Use MATLAB to obtain graphs for all four spring/mass cases.



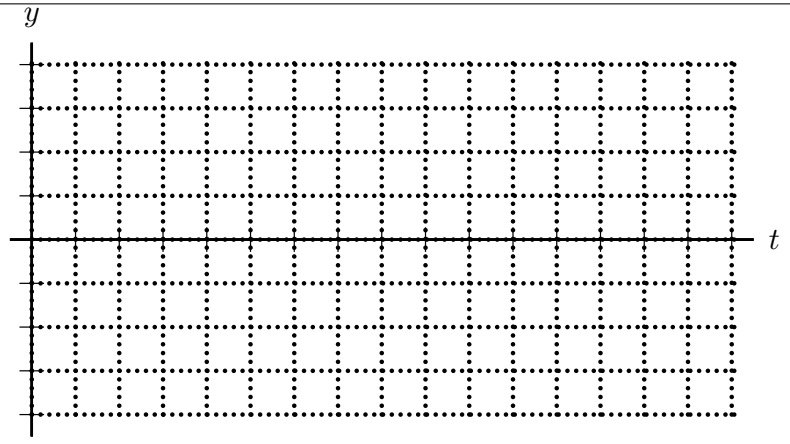
Undamped



Under-Damped



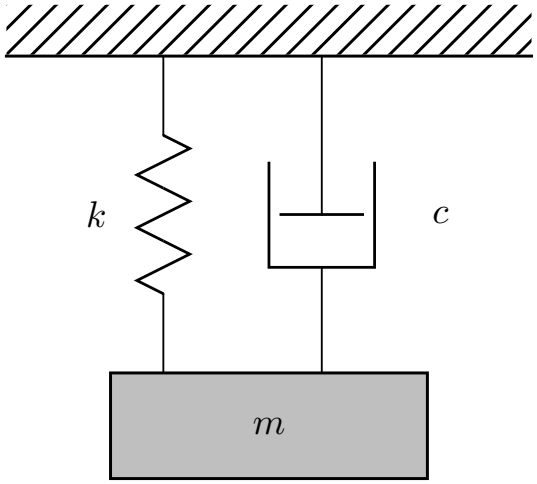
Critically Damped



Over-Damped

Spring/Mass with Periodic External Forces

We now add another component to our spring system model: **external forces**.



Problem. Write out the DE for the position of the mass, given the addition now of an external force, given by $F_{\text{ext}} = A\sin(Bt)$.

What are analagous systems to this that you might have encountered before?

Problem. Write a pair of MATLAB scripts and functions that will let you simulate the motion of the mass given an external force of the form $F_{\text{ext}} = A\sin(Bt)$.

$$mx'' = -kx - cx' + A\sin(Bt)$$

Once you have the MATLAB code written, we can explore the solutions of/predictions for the spring/mass system for different scenarios, by changing parameters in our MATLAB script.

Simulations of the Forced Spring/Mass System

For the following systems, use $m = 1$ kg and $k = 25$ N/m, unless otherwise specified.

Unforced Motion

Problem. Setting $F_{\text{ext}} = 0$, review the effect of increasing the damping coefficient c on the behaviour of the mass.

Forced, but Undamped Motion

Now we turn off the damping (by setting $c = 0$) to zero, and start to apply the external force.
We set $F_{\text{ext}} = 10\sin(1.0t)$.

Experiment with the frequency in F_{ext}, in the range 0.5 to 3, or well above 5. Does the motion of the mass over time make sense?

Resonance

In unforced, undamped systems, there is a **natural frequency** for the mass, given by $\omega = \sqrt{\frac{k}{m}}$.

Problem. Set $F_{\text{ext}} = 1\sin(5t)$. What is special about the frequency 5 rad/s?

Describe the graph of the solution.

Give the physical reasons for the graph seen in the simulation.

The condition of having the external force at *exactly* the same frequency as the natural vibrations is called **resonance.** Note: for true resonance, there must be *no damping.*

Near-Resonance

Admitedly, resonance is a unique event requiring **perfect** matching of the stimulating force with the natural frequency.

Problem. Explore frequencies in F_{ext} *close to* 5. Describe the resulting solutions, both for their amplitude and any other response.

Give a physical reason for the shape of the graph for near-resonance response.

Note again that there is *no damping* with this response.

Adding in Friction - Practical Resonsance

Set $F_{\text{ext}} = 1\sin(5t)$ again, but set damping c to 0.1. **Problem.** Compare this solution to the one without damping.

Gradually increase the amount of damping. Describe how the solution changes.

How is this more realistic than the undamped case?

How do the mathematical forms of the pure-resonance (undamped) and practical resonance (some damping) compare?

Transient and Steady-State Solutions

More generally, when there is damping in the spring/mass system and an oscillatory external force, we can break the solution into two parts: *transient* and *steady state*.
We set $F_{\text{ext}} = 1\sin(t)$, and $c = 1$ for damping, which is fairly high for an oscillating system. Describe the resulting behaviour.

Specifically, what are the two natural frequencies in the solution?

Which frequency is present in the **long run**?

Reminder: we study the simple damped spring/mass system in such depth because

- the system displays all the interesting mathematical solution forms as we vary just a few parameters,
- we hope that the simplicity of the system, and its resemblance to familiar systems like swings or bouncing a basket ball, help to you associate the mathematics with the real-world behaviour.
- the DE for the spring/mass system is either identical or similar to those for a surprising number of other real-world scenarios.

Classifying Higher-Order DEs

While working with DEs in MATLAB, we haven't had to distinguish between types of equations: so long as they can be re-written in the form $\frac{dy}{dt} = f(t, y)$, or the vector form $\frac{d\vec{w}}{dt} = f(t, \vec{w})$, MATLAB can produce a numerical solution or prediction.

However, when looking at references or mathematical derivations, where DEs are solved exactly using by-hand methods, the method used is usually chosen based on the form of the DE, so being able to categorize equations is a helpful skill.

Problem. Distinguish between the following classifications of differential equations, and given an example of each.

Linear vs Non-Linear

Linear Homogeneous and Linear Non-Homogeneous

Classify the following DEs based on the terms *order*, *homogeneous* and *linear*.

$$x^2y'' + xy' + y = 10$$

$$100y'' + y = 4x^3$$

$$(y'')^3 + y = 4e^x$$

$$4y^{(4)} - 10y' + y = 0$$