# General Second-Order Runge-Kutta — Forced Oscillation

Analyzed in class, February 7, 2025

This is the fifth notebook for you to complete. Get the **rungeKutta2[]** implementation working before class, and then we will analyze it in class together. By the way, a synonym for forced oscillation is “driven oscillation.”

## Forced Oscillation

### Problem Description

In[34]:=
```
omega1 = 1;
externalForce[t_] := 100 Sin[omega1 t]
springConstant = 20;
dampingConstant = 1;
force[t_, x_, v_] := -springConstant x - dampingConstant v + externalForce[t]
mass = 5;
a[t_, x_, v_] := force[t, x, v] / mass;
tInitial = 0;
tFinal = 100;
steps = 25 000;
deltaT = (tFinal - tInitial) / steps;
```

### Initial Conditions

Let’s let the spring be initially unstretched with no velocity and see what the external force does to it:

In[45]:=
```
xInitial = 0.0;
vInitial = 0.0;
(* We also want to be able to visualize the external force,
so let's tack it on to the quantities that will be tabulated by NestList *)
initialConditions =
  {tInitial, xInitial, vInitial, externalForce[xInitial] / springConstant};
```

### General Second-Order Runge-Kutta — Implementation

The implementation will be almost the same as in the damped oscillation notebook you completed last Friday.

There are only a few small things that have to be changed. Figure out what they are and then if you still have that code handy, you can almost completely re-use what you did in that notebook. Or just write it out again! The more times you write it out, the better you will remember it.

```
lambda = 1; (* I am going to switch to a different letter for
 what we have been calling alpha and you should too. We are running
 out of common letters and I will want to reuse alpha for something
 else in the next notebook which is the pendulum problem. *)
rungeKutta2[cc_] := (
  (* Extract time, position, and velocity from the list *)
  curTime = cc[[1]];
  curPosition = cc[[2]];
  curVelocity = cc[[3]];
  (* Compute tStar, xStar, vStar *)
  tStar = curTime + lambda deltaT;
  (* Implement General Second-Order Runge-Kutta *)
  newTime = curTime + deltaT;
  (* We are keeping track of the contribution
   to the acceleration due to the external force *)
  {newTime, newPosition, newVelocity, externalForce[newTime] / springConstant}
 )
N[rungeKutta2[initialConditions]] (* Test your rungeKutta2 function. *)
(* The output just below should be {0.004, 3.19999*10^-7, 0.00016, 0.0199999}. *)
```

## Displaying Forced Oscillation

Nest the procedure and then transpose the results to produce position and velocity plots:

In[51]:=
```
rk2Results = NestList[rungeKutta2, initialConditions, steps];
rk2ResultsTransposed = Transpose[rk2Results];
positionPlot = ListPlot[Transpose[rk2ResultsTransposed[[{1, 2}]]]]
```

In[53]:=
```
positions = rk2ResultsTransposed[[2]];
forces = rk2ResultsTransposed[[4]];
```

In[55]:=
```
Animate[NumberLinePlot[{positions[[step]], forces[[step]]}, PlotRange → {-50, 50}],
 {step, 0, steps, 1}, DefaultDuration → 20]
```

## Conclusion / Commentary

Our oscillator now has the force law $F(x) = -20\,x - v$ *and in addition a sinusoidal external driving force.* You will remember that in the conclusion last Friday's notebook (before the epilog on car suspension) I defined $\omega_0$ and $\gamma$. In this notebook, we additionally have the driving frequency **omega1**. So now you see why I put the subscript "0" on $\omega$ in the previous notebook. In total we now have three relevant frequencies in the damped, driven harmonic oscillator.

| | |
|---|---|
| $\omega_1$ | the external or driving frequency |
| $\omega_0$ | the natural frequency of the oscillator in the absence of damping |
| $\gamma$ | the frequency that controls the rate of decay of the exponential |

There is enough complexity in this system that it provides a lot to play around with. Once you all have it working, we'll adjust the input parameters to get other values for the frequencies.

### Correcting a Bit of Sloppiness

There is actually a fourth frequency which is derived from two others:

$\sqrt{\omega_0^2 - \gamma^2}$ the frequency of the oscillator including damping (which slows it down a little)

It didn't occur to me to mention this frequency previously, because it makes such a small difference that I had actually forgotten about it. Also, I have never shown you the full solution, let alone derived it, which requires more differentiation and algebra than we typically do in this course. To see how small a difference it makes with our values for $\omega_0$ and $\gamma$, let's look at how much this ratio differs from 1:

$$\frac{\sqrt{\omega_0^2-\gamma^2}}{\omega_0} = \sqrt{1-\frac{\gamma^2}{\omega_0^2}} \approx 1-\frac{1}{2}\frac{\gamma^2}{\omega_0^2} = 1-\frac{1}{2}\frac{(b/2\,m)^2}{k/m} = 1-\frac{1}{8}\frac{b^2}{km} = 1-\frac{1}{8}\frac{1^2}{20\cdot 5} = 1-\frac{1}{800} = 0.99875 \approx 0.999$$

So it is about a one-part-in-a-thousand effect. Still, the effect piles up over many oscillations. If you were to do 800 oscillations using $\omega_0$, you'd discover that the actual system which has $\sqrt{\omega_0^2 - \gamma^2}$ as its oscillation frequency had only done 799.

## Epilog — Resonance

Let's display $\omega_1$, $\omega_0$, and $\gamma$ for easy comparison:

In[56]:=
```
omega0 = Sqrt[springConstant / mass];
gamma = dampingConstant / (2 mass);
{omega1, omega0, gamma}
```

At minimum, one of the things we should do as part of playing around with this system is just to jack up $\omega_1$ so that it is almost as large as $\omega_0$. For example $\omega_1 = 1.8$ is good to try. That is "just below resonance."

Then put $\omega_1$ right on resonance (in the absence of damping): $\omega_1 = 2$. When the system is driven very near resonance, it oscillates widely. It will help to make a smoother and easier-to-contemplate animation to jack up the number of steps to 50 000 and also to slow the animation duration down by doubling the animation duration: `DefaultDuration → 40`.

Then we should try $\omega_1 = 2.2$. That is "just above resonance."

Seeing real systems being driven near resonance was the traditional way of showing what a modern Mathematica notebook like this one shows before computer simulations reduced the incremental value of real lecture demonstrations. I still find seeing videos of real systems helpful. Here is a good one showing resonance: https://youtu.be/aZNnwQ8HJHU.