Torsion Pendulum

Completed and Analyzed in class, March 18, 2025

This is the eleventh notebook for you to complete. It bears strong similarity to our third notebook (Mass on a Spring).

The only significant difference is that a mass on a spring usually has its position measured by its displacement x from equilibrium, whereas a torsion pendulum has its angle θ, measured by its rotation from equilibrium.

Torque and Angular Acceleration

Torsion is what we might more casually call "twist." When a material is twisted it generally opposes the twist. If you twist something limp, like a rope candy, it doesn't oppose the twist very much, and it may even deform so that it stops opposing the twist at all . But if you twist a thin stainless steel wire, it will oppose the twist, and unless you twist it so hard that you damage it, it will oppose the twist until it has returned to its original state. The opposition to the twist is an example of "torque."

We have used Newton's Second Law, $F = ma$, a lot of times. There is a similar formula $\tau = I\alpha$, that we need to use for the torsion pendulum. Newton's law is read, "force (F) equals the mass (m) times the acceleration (a)." The new formula involving torque is read, "torque (τ) equals the moment of inertia (I) times angular acceleration (α)." I want to mention and stress that this is not an independent law. In a mechanics course, you would derive this law from Newton's Second Law.

So we suppose we have a rod with a moment of inertia, I, and it is connected to two fixed walls by two thin stainless wires. The wires oppose the twist. The amount they oppose it is defined by some constant κ and the amount of twist.

Here is the formula:

$$\tau = -\kappa_L \theta - \kappa_R \theta$$

We combine this with $\tau = I\alpha$ to get:

$$\alpha = (-\kappa_L \theta - \kappa_R \theta) / I$$

In[•]:=
```
momentOfInertia = 0.1;
kappaLeft = 0.5;
kappaRight = 0.5;
omega0 = Sqrt[(kappaLeft + kappaRight) / momentOfInertia];
α[theta_] := -omega0^2 theta;
```

Graphics

I left work for you. See the comment in the code for where your work goes. It is going to involve halfHeight, and sines and cosines of angle.

```
halfHeight = 1;
halfDepth = 1;
halfWidth = 5;
region = {FaceForm[{Blue, Opacity[0.1]}], Cuboid[
    {-halfWidth, -halfDepth, -halfHeight}, {halfWidth, halfDepth, halfHeight}]};
leftWire = {Red, Line[{{-halfWidth, 0, 0}, {0, 0, 0}}]};
rightWire = {Green, Line[{{0, 0, 0}, {halfWidth, 0, 0}}]};
torsionRodGraphic[angle_] := Graphics3D[{
   region,
   leftWire,
   rightWire,
   (* The rod does not yet rotate. Make it rotate by angle. *)
   {Black, Thick, Line[{{0, -halfDepth , 0.0}, {0, halfDepth , 0.0}}]}
  }]
torsionRodGraphic[20 °]
```

Simulation Parameters

In[•]:=
```
tInitial = 0.0;
numberOfPeriods = 10;
period = 2 Pi / omega0;
tFinal = numberOfPeriods period;
steps = 1000 numberOfPeriods;
deltaT = (tFinal - tInitial) / steps;
```

Initial Angle and Angular Velocity

Let's let the pendulum be initially held still at 20° and released:

In[•]:=
```
tInitial = 0.0;
thetaInitial = 20 °;
omegaInitial = 0;
initialConditions = {tInitial, thetaInitial, omegaInitial};
```

Second-Order Runge-Kutta — Theory

$$t^* = t + \frac{\Delta t}{2}$$

$$\theta^* = \theta(t_i) + \omega(t_i) \cdot \frac{\Delta t}{2}$$

$$t_{i+1} = t_i + \Delta t$$

$$\omega(t_{i+1}) = \omega(t_i) + \alpha(\theta^*) \cdot \Delta t$$

$$\theta(t_{i+1}) = \theta(t_i) + (\omega(t_i) + \omega(t_{i+1})) \frac{\Delta t}{2}$$

Second-Order Runge-Kutta — Implementation

As usual, lots for you to finish.

```
rungeKutta2[cc_] := (
  (* Extract time, angle, and angular velocity from the list *)
  curTime = cc[[1]];
  curTheta = cathedral;
  curOmega = bridge;
  (* Compute tStar, and thetaStar *)
  tStar = sandal;
  thetaStar = shoe;
  (* Compute newTime, newTheta, and newOmega *)
  newTime = rocks;
  newOmega = water;
  newTheta = plants;
  {newTime, newTheta, newOmega}
 )
N[rungeKutta2[initialConditions]]
(* Test the rungeKutta2 function you just completed. *)
(* The output just below should be {0.00198692,0.349059,-0.006935} *)
```

Displaying Oscillation as a Graph

Nest the procedure, transpose the results, and produce a plot of the angle θ as a function of time:

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

Animating the 3D Graphics

It's also nice to have an animation, arranged so that the default duration of the animation is the actual duration of the animation:

In[ ]:=
```
Animate[torsionRodGraphic[thetas[[step]]],
 {step, 0, steps, 1}, DefaultDuration → tFinal - tInitial]
```