Double Pendulum

Completed and Analyzed in class, February 25, 2025

This is the tenth notebook for you to complete.

Double Pendulum — Angular Accelerations — Recap

Copied over from the theory we just presented (without derivation):

$$\alpha_1 = -\tfrac{1}{16}\cos(\theta_2 - \theta_1) + \tfrac{1}{16}\,\omega_2{}^2 \sin(\theta_2 - \theta_1) - 4\,\pi^2 \sin\theta_1$$

$$\alpha_2 = -4\,\alpha_1 \cos(\theta_2 - \theta_1) - 4\,\omega_1{}^2 \sin(\theta_2 - \theta_1) - 16\,\pi^2 \sin\theta_2$$

Well, that is easy enough to code up:

In[ ]:=

```
alpha1[theta1_, theta2_, omega1_, omega2_] :=
  - 1/16 Cos[theta2 - theta1] + 1/16 omega2^2 Sin[theta2 - theta1] - 4 Pi^2 Sin[theta1];

alpha2[theta1_, theta2_, omega1_, omega2_] :=
  -4 alpha1[theta1, omega1, theta2, omega2] Cos[theta2 - theta1] -
   4 omega1^2 Sin[theta2 - theta1] - 16 Pi^2 Sin[theta2];
```

Initial Conditions

First set up the duration. Let’s also define **`steps`** and **`deltaT`** while we are at it:

In[ ]:=

```
tInitial = 0.0;
tFinal = 120.0;
steps = 7200;
deltaT = (tFinal - tInitial) / steps;
```

We’ll pull longer pendulum back to -45° and the shorter one we’ll just let hang straight down:

In[ ]:=

```
theta1Initial = -45 °;
theta2Initial = 0 °;
omega1Initial = 0.0;
omega2Initial = 0.0;
```

In[ ]:=
```
initialConditions =
  {tInitial, theta1Initial, theta2Initial, omega1Initial, omega2Initial}
```

Out[ ]=
```
{0., -45 °, 0, 0., 0.}
```

Second-Order Runge-Kutta — Double Pendulum — Recap

Also copied over from the theory:

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

$\theta_1{}^* = \theta_1(t_i) + \omega_1(t_i) \cdot \frac{\Delta t}{2}$

$\theta_2{}^* = \theta_2(t_i) + \omega_2(t_i) \cdot \frac{\Delta t}{2}$

$\omega_1{}^* = \omega_1(t_i) + \alpha_1(\theta_1, \omega_1, \theta_2, \omega_2) \cdot \frac{\Delta t}{2}$

$\omega_2{}^* = \omega_2(t_i) + \alpha_2(\theta_1, \omega_1, \theta_2, \omega_2) \cdot \frac{\Delta t}{2}$

$\omega_1(t_{i+1}) = \omega_1(t_i) + \alpha_1(\theta_1{}^*, \omega_1{}^*, \theta_2{}^*, \omega_2{}^*) \cdot \Delta t$

$\omega_2(t_{i+1}) = \omega_2(t_i) + \alpha_2(\theta_1{}^*, \omega_1{}^*, \theta_2{}^*, \omega_2{}^*) \cdot \Delta t$

$\theta_1(t_{i+1}) = \theta_1(t_i) + (\omega_1(t_i) + \omega_1(t_{i+1})) \frac{\Delta t}{2}$

$\theta_2(t_{i+1}) = \theta_2(t_i) + (\omega_2(t_i) + \omega_2(t_{i+1})) \frac{\Delta t}{2}$

Second-Order Runge-Kutta — Implementation

In[ ]:=
```
rungeKutta2[cc_] := (
  curTime = cc[[1]];
  curTheta1 = cc[[2]];
  curTheta2 = cc[[3]];
  curOmega1 = cc[[4]];
  curOmega2 = cc[[5]];
  newTime = curTime + deltaT;
  theta1Star = curTheta1 + curOmega1 deltaT / 2;
  theta2Star = curTheta2 + curOmega2 deltaT / 2;
  omega1Star =
   curOmega1 + alpha1[curTheta1, curTheta2, curOmega1, curOmega2] deltaT / 2;
  omega2Star =
   curOmega2 + alpha2[curTheta1, curTheta2, curOmega1, curOmega2] deltaT / 2;
  newOmega1 =
   curOmega1 + alpha1[theta1Star, theta2Star, omega1Star, omega2Star] deltaT;
  newOmega2 =
   curOmega2 + alpha2[theta1Star, theta2Star, omega1Star, omega2Star] deltaT;
  newTheta1 = curTheta1 + (curOmega1 + newOmega1) deltaT / 2;
  newTheta2 = curTheta2 + (curOmega2 + newOmega2) deltaT / 2;
  {newTime, newTheta1, newTheta2, newOmega1, newOmega2}
 )
rungeKutta2[initialConditions]
```

Out[ ]=

```
{0.0166667, -0.781525, -0.0109835, 0.464839, -1.31802}
```

Using NestList[] to Repeatedly Apply rungeKutta2[]

In[ ]:=
```
rk2Results = Transpose[NestList[rungeKutta2, initialConditions, steps]]
```

Out[ ]=

```
{ … 1 … }
```

Full expression not available (original memory size: 2.9 MB)

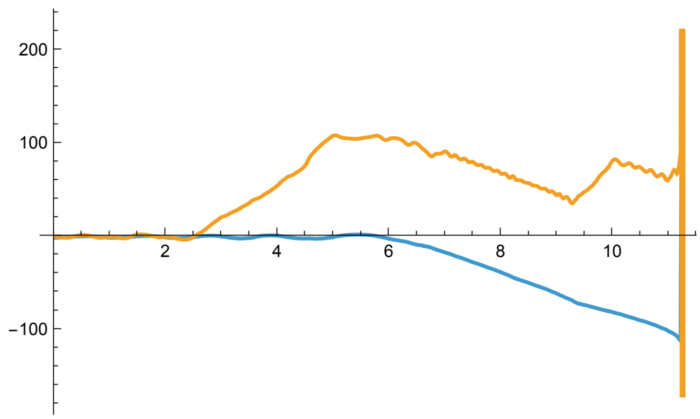
Transposing to Get Points We Can Put in ListLinePlot[]

In[ ]:=
```
times = rk2Results[[1]];
theta1s = rk2Results[[2]];
theta2s = rk2Results[[3]];
```

In[ ]:=
```
times
```
Out[ ]=
```
{0., 0.0166667, 0.0333333, 0.05, 0.0666667, 0.0833333, 0.1, 0.116667, 0.133333, 0.15,
 … 7181 …, 119.85, 119.867, 119.883, 119.9, 119.917, 119.933, 119.95, 119.967, 119.983, 120.}
```
Full expression not available (original memory size: 174.6 kB)

In[ ]:=
```
timesAndTheta1s = Transpose[{times, theta1s}];
timesAndTheta2s = Transpose[{times, theta2s}];
```
In[ ]:=
```
ListLinePlot[{timesAndTheta1s, timesAndTheta2s}]
```
ListLinePlot: Value of option PlotRange ->
{{0, 120.}, {$-2.374004556376794\times10^{622}$, $5.452850347280646\times10^{622}$}} is not All, Full, Automatic, a positive machine number, or an appropriate list of range specifications.

Out[ ]=



A Graphic

The graphics work is straightforward but a little time-consuming, and not terribly instructive, so it is already all done:

In[ ]:=
```
coupledOscillatorGraphic[positionList_] := Graphics[{
    width = 10;
    buffer = 0.5;
    netWidth = width - 2 buffer;
    wallHeight = 1.0;
    numberOfSprings = n + 1;
    (* the next line makes a gray rectangle *)
    {EdgeForm[Thin], Gray, Polygon[{{0, -1}, {width, -1}, {width, 1}, {0, 1}}]},
    (* the next two lines make the walls *)
    Line[{{buffer, -wallHeight / 2}, {buffer, wallHeight / 2}}],
    Line[{{width - buffer, -wallHeight / 2}, {width - buffer, wallHeight / 2}}],
    (* finally we draw all the points *)
    Table[Style[Point[{positionList[[j]] + netWidth/numberOfSprings j + buffer, 0.0}],
      PointSize[0.03]], {j, n}]
  }]
(* A little test to see if our code at least draws equally-
 spaced points when the positions are all zero: *)
coupledOscillatorGraphic[Table[0.0, n]]
```

Table: Non–list iterator n at position 2 does not evaluate to a real numeric value.

Table: Iterator {j, n} does not have appropriate bounds.

Out[ ]=

Animating The Graphics