Forced Pendulum — Phase Space — Chaos

Done in class, February 11, 2025.

This is our seventh numerical methods notebook.

Forced Oscillation

Angular Acceleration α

In[28]:=
```
mass = 5;
gravity = 9.80665;
(* the value of gravity in units of meters / seconds-squared *)
length = 0.24840;
(* A pendulum whose length is 9.7795 inches converted to meters  *)
(* The natural frequency of such a
 pendulum provided the swings are not large: *)
omega0 = Sqrt[gravity / length];
gamma = 0.03;
(* A real pendulum swinging in air typically has a small gamma. *)
period = 2 Pi / omega0;
(* The length was chosen so that the period is 1 second. To be *)
(* precise, 2 Pi / omega0 = 0.999989,
and 2 Pi / Sqrt[omega0^2-gamma^2] = 1.000000. *)
omega1 = 0.7 * 2 Pi;
drivingForce[t_] := length 10 Sin[omega1 t]; (* tangential driving force *)
α[t_, theta_, omega_] :=
  -omega0^2 Sin[theta] -2 gamma omega +drivingForce[t] / (length mass);
```

Simulation Parameters

In[37]:=
```
tInitial = 0.0;
tFinal = 50.0;
steps = 200 000;
deltaT = (tFinal - tInitial) / steps;
```

Initial Angle and Angular Velocity

Let's let the pendulum be initially at rest and see what the driving force does to it:

In[41]:=
```
thetaInitial = 0;
omegaInitial = 0;
initialConditions = {tInitial, thetaInitial, omegaInitial};
```

General Second-Order Runge-Kutta — Implementation

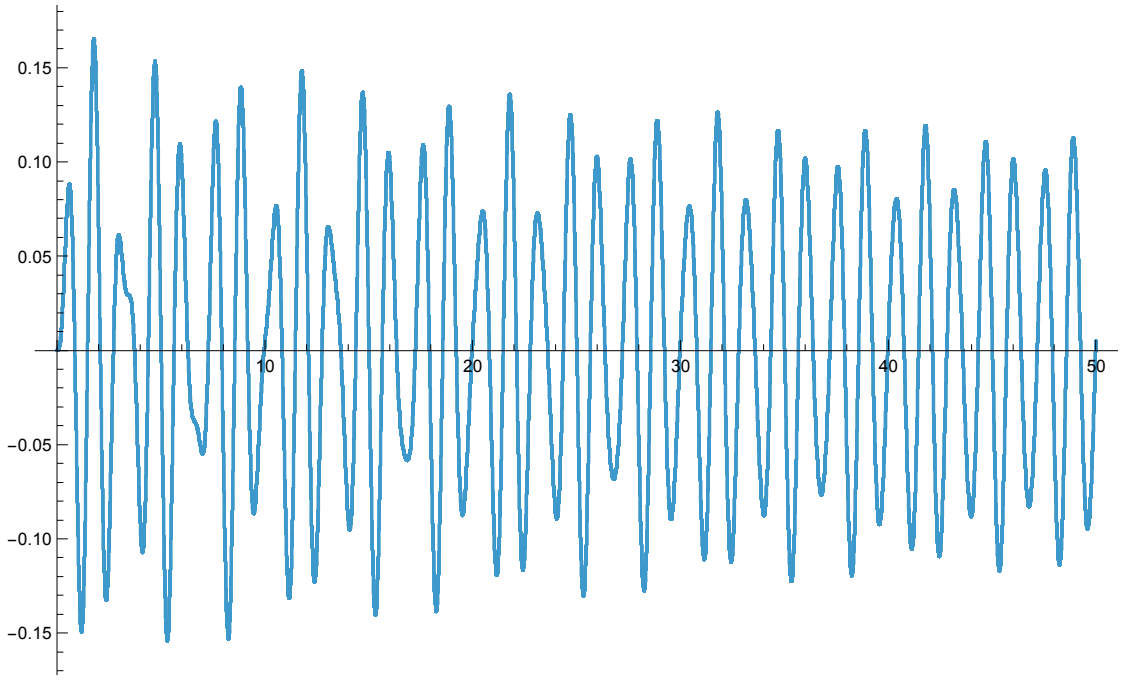
In[44]:=

```
lambda = 1;
rungeKutta2[cc_] := (
  (* Extract time, angle, and angular velocity from the list *)
  curTime = cc[[1]];
  curAngle = cc[[2]];
  curAngularVelocity = cc[[3]];
  (* Compute tStar, xStar, vStar *)
  tStar = curTime + lambda deltaT;
  thetaStar = curAngle + curAngularVelocity lambda deltaT;
  omegaStar =
   curAngularVelocity + α[curTime, curAngle, curAngularVelocity] lambda deltaT;
  (* Implement General Second-Order Runge-Kutta *)
  newTime = curTime + deltaT;
  newAngularVelocity =
   curAngularVelocity + ((1 - 1/(2 lambda)) α[curTime, curAngle, curAngularVelocity] +
       1/(2 lambda) α[tStar, thetaStar, omegaStar]) deltaT;
  newAngle = curAngle + (curAngularVelocity + newAngularVelocity) deltaT / 2;
  {newTime, newAngle, newAngularVelocity}
 )
```

Computing and Displaying Angle as a Function of Time

In[46]:=
```
rk2Results = NestList[rungeKutta2, initialConditions, steps];
rk2ResultsTransposed = Transpose[rk2Results];
times = rk2ResultsTransposed[[1]];
thetas = rk2ResultsTransposed[[2]];
omegas = rk2ResultsTransposed[[3]];
thetaPlot = ListPlot[Transpose[{times, thetas}]]
```

Out[51]=



Studying the plot of angle as a function of time, it may not be obvious that the system has become chaotic.

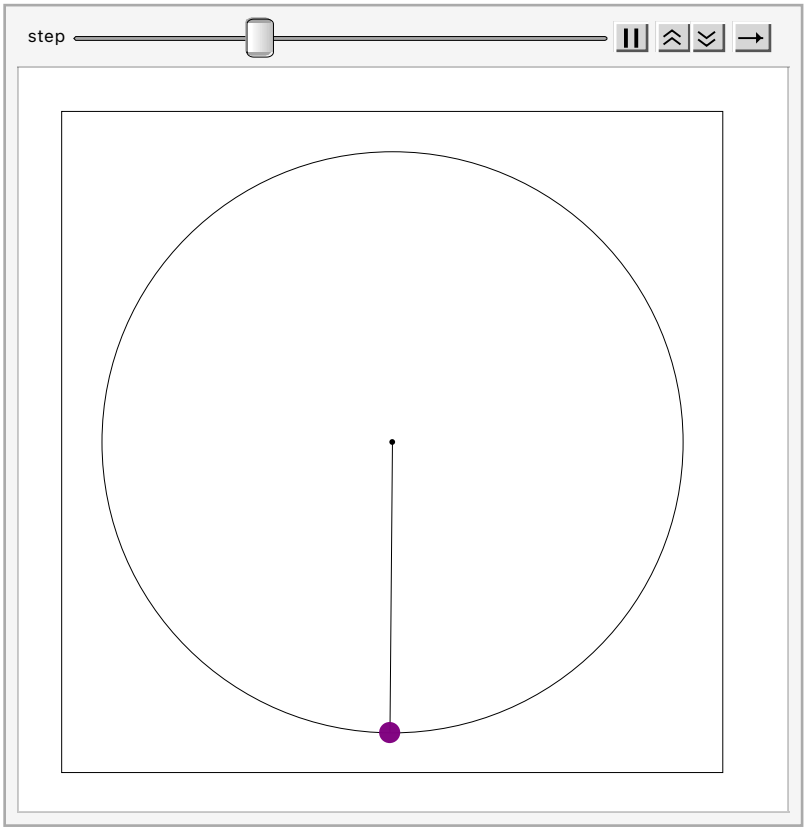
Animating the Pendulum

Neither is it obvious watching the animation, although it does look screwy. Perhaps it is repeating, but just in some complicated way that isn't obvious?

In[52]:=
```
pendulumGraphic[angle_] := Graphics[{
    EdgeForm[Thin], White,
    RegularPolygon[{0.0, 0.0}, 0.4, 4],
    Black,
    Circle[{0, 0}, length],
    Point[{0, 0}],
    Line[{{0, 0}, length { Sin[angle], - Cos[angle]}}],
    PointSize[0.03], Purple,
    Point[length { Sin[angle], - Cos[angle]}]
   }];
```

In[53]:=
```
Animate[pendulumGraphic[thetas[[step]]], {step, 0, steps, 1}, DefaultDuration → tFinal - tInitial]
```

Out[53]=



Phase Space Plot — Chaos

People have decided that phase space plots are a good way to convince each other that a system is not settling down or repeating. On the horizontal axis of the pendulum phase space plot is the angle. On the vertical axis is angular velocity. Time is not shown, although you can see where the plot below starts (at the origin).

To convince you that a phase space plot like this is quite out of the ordinary compared to anything we have previously studied, it would be good to go back to the forced harmonic oscillator notebook and add a phase space plot to it. You will quickly see that a phase space plot in a system that is settling down to a periodic motion looks far more regular.

In[54]:=
```
phaseSpacePlot = ListPlot[Transpose[{thetas, omegas}]]
```

Out[54]=