EXERCISES

Mathematica 6 ~ Lab Number 4

Problem 1. Think for a moment about how you would proceed to solve *by hand* the linear first-order ODE

$$y\,'(x) = -\cos x \cdot y(x)$$

Use **DSolve** to ask *Mathematica* to do the same thing. The answer will contain a single constant of integration, called `C[1]`. The following command **somesolutions=Table[** place general solution here **/.C[1]->i,{*i*, −5, 5}]** produces a list of particular solutions. Plot **somesolutions** on the interval $0 < x < 4\pi$. Place ticks at the points $\{0, \pi, 2\pi, 3\pi, 4\pi\}$.

Problem 2. Lord Rayleigh was led from the physics of violin strings to a nonlinear ODE of the form

$$x''(t) + \Big\{\tfrac{1}{3}\big[x'(t)\big]^2 - 1\Big\}x'(t) + x(t) = 0$$

The command **DSolve** informs us that no analytical solution exists; we must use **NDSolve**. Assign names

> **violin1** to the solution that proceeds from $x(0) = 0.1, x'(0) = 0$;
> **violin2** to the solution that proceeds from $x(0) = 1.0, x'(0) = 0$;
> **violin3** to the solution that proceeds from $x(0) = 1.9, x'(0) = 0$.

Working on the interval $0 < t < 15$, construct a figure that superimposes the numerical solutions **violin1**, **violin1**, and **violin1**, using the option **PlotStyle->{Red,Blue,Black}** to assign to those respective curves the colors just indicated.

Problem 3. The simple oscillator equation reads $mx'' + kx = 0$ if the spring is "perfect," but if the spring is what engineers call "hard" then

$$k \text{ constant} \longrightarrow k + ax^2$$

in better approximation (a is a phenomenological constant, and refers to a second property of the spring). The equation of motion has become

$$mx''(t) + kx(t) + ax^3(t) = 0$$

Set $m = 1$, $k = 0.3$, $a = 0.04$, $x(0) = 0$ and

$$x'(0) = 1, 2, 3, 4, 5$$

Working again on the interval $0 < t < 15$, and on the pattern rehearsed in the preceding problem, give names **`hardspring1`** through **`hardspring5`** to the five numerical solutions supplied by **`NDSolve`**, then plot those solutions. To better distinguish one from another, introduce the definition
`codecolor={Blue, Green, Orange, Red, Black}`
and install the option **`PlotStyle->codecolor`**. Notice that increasing the initial velocity (energy) cause the particle to venture further into the stiff spring domain, and decreases the period.

The **`Manipulate`** provides a more fluid way to display such information. Define **`hardspring[v_]:=x[t]/.First[NDSolve[{`***the differential equation and boundary conditions, except that initial velocity has been identified with an adjustable parameter* **`v},x[t], {t,0,15}]]`**, then manipulate the figures
`Plot[Evaluate[hardspring[v]], {t,0,15}, PlotRange→ {-6,6}, PlotStyle→ {Red, Thick}]` as v ranges $1 \leqslant v \leqslant 5$.

Problem 4. To model an oscillator in which the spring "gets tired" we might write

$$mx''(t) + ke^{-t/\tau}x(t) = 0$$

We will look to the case $m = 1$, $k = \tau = 4$. While *Mathematica* 5.2 was obliged in this instance to proceed numerically, *Mathematica* 6 is happy to provide analytical solutions. Set $x'(0) = 0$, $x(0) = 1, 2, 3, 4, 5$. Call the solutions **`tired1`** through **`tired5`**. Plot the solutions reported by **`DSolve`**, using **`colorcode`** once again to distinguish one curve from another. Notice that the zero-crossings are energy independent, but are spaced farther and farther apart as the spring loses its poop.

Problem 5. To "pump" a swing we use our internal energy reserves to manipulate a system parameter (that that instance, the *effective length* of the rope), thus creating an instance of a "parametrically stimulated oscillator." We are careful to pump in synchrony with the natural period of the swing, since our objective is to pump energy into the system. The equation

$$x''(t) + \{1 + 0.2\sin 2t\}x(t) = 0$$

serves to illustrate the process. Construct
`stimulated=x[t]/.NDSolve[{x''[t]+(1+0.2Sin[2t])x[t]==0, x'[0]==0, x[0]==1},x[t],{t,0,50}]`
and
`unstimulated=` same thing with **`0.2`** → **`0.0`**
and then construct a figure showing superimposed graphs of those functions. You will want to install the option
`PlotRange->All`
To demonstrate the importance of synchrony, construct
`outofsync=` same as **`stimulated`** but with **`2`** → **`2.2`**
and a figure in which graphs of **`outofsync`** and **`unstimulated`** are superimposed.