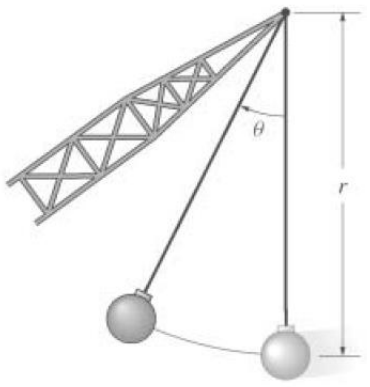
Ordinary Differential Equations: Pendulum Problems

MATH2070: Numerical Methods in Scientific Computing I

Location: http://people.sc.fsu.edu/~jburkardt/classes/math2070_2019/odes_pendulum/odes_pendulum.pdf



Pendulum problems can involve many modeling choices.

Pendulum problems

Consider various models of the pendulum problem, the effect of stepsize and initial conditions, the modeling on nonlinear effects and damping, and the use of phase plots to analyze the solution.

1 Flash back: Phase plot analysis for the predator problem

The predator problem was introduced in the previous lab as a test problem and you were asked to plot the rabbit and fox populations over time.

$$\begin{aligned} r'(t) &= 2r - 0.001 * r * f \\ f'(t) &= -10f + 0.002 * r * f \\ r(0.0) &= 5000 \\ f(0.0) &= 100 \end{aligned}$$

We can investigate this problem with the code *predator_euler(n)*, where we are allowed to specify the number of time steps as an input argument.

We are aware that the results produced by the Euler method are approximations. For our experiments, we take n equal steps between a starting and final time. We know that accuracy depends in part on the stepsize, and so we want to feel confident that we have picked a number of steps large enough that our approximation is reasonable.

How can we do this when we are solving a problem whose exact answer is unknown? Sometimes, even if we don't know the true solution, we have certain expectations of the result, such as that it must stay positive, or be bounded, or maintain a fixed energy. In such cases, comparing the results of two values of n, or looking at plots, can be useful in judging our results.

For our predator problem, using a small value of $n = 100$, the computation blows up. Increasing n to 500 or 1000 can give a bounded solution that seems to oscillate and grow. Is this growth a true property of the system, or is it really just a symptom of a weak approximation? As we did for the pendulum problem, we can try an even larger value of n, at which point it seems that we are getting a periodic solution.

Because the right hand sides do not depend on time, we can get a lot of information from a phase plot, which in this case maps $r(t)$ against $f(t)$, or in general, $y(t)$ against $y'(t)$. The phase plot of the data is more convincing, and should suggest that an exact solution of this problem would be periodic.

In fact, this kind of predator-prey problem (also called a Lotka-Volterra equation) has a conserved quantity, somewhat like the energy in a physical system. For our problem, this quantity has the form

$$E(r, f) = 0.002f + 0.001r - 10\ln(r) - 2\ln(f)$$

and you can convince yourself, using a little algebra, that

$$\frac{dE(r,f)}{dt} = \frac{\partial E}{\partial r}\frac{dr}{dt} + \frac{\partial E}{\partial f}\frac{df}{dt} = 0$$

Hence, the phase plot is tracing out contour lines $E =$ constant.

Note that every point in this plot represents a pair of values that can be used to specify the initial value for an ODE problem. In this case, these would be the initial number of rabbits and foxes.

Also, note that if we compute several solutions with differing initial conditions, we will get a nested sequence of curves, one inside the other. These curves can't cross, because that would mean that two different solutions share a common point. If we used that common point as the initial condition, we can't get two different solution curves.

An interesting feature of this predator problem is that it has a fixed point, that is, there is a pair of positive values of r and f for which the solution over time does not change at all. We can determine these values by assuming that neither r nor f is zero, and then solving

$$\begin{aligned} \frac{dr}{dt} &= 2r - 0.001 * r * f = 0 && \rightarrow f = 2000 \\ \frac{df}{dt} &= -10f + 0.002 * r * f && \rightarrow r = 500 \end{aligned}$$

2 The pendulum problem

We have previously mentioned the linear pendulum problem, which models the variation in θ, the angle made by a pendulum, as measured counterclockwise from the downward vertical position. Here, θ' is the angular velocity, and θ" the angular acceleration. The pendulum has a length l, and a mass m and is subject to a gravitational force vector which points downward with magnitude g.

Newton's law relates force and acceleration by $F = ma$. Because the pendulum is constrained on a string, it can only move in the angular direction. Therefore, its motion is only in response to the projection of gravity in that direction, which is $-g\sin(\theta)$. The position s of the pendulum along the circle of radius l is related to θ by $s = l\,\theta$. Thus, $F = ma$ for this problem becomes:

$$\begin{aligned}
F &= m\,a \\
-m\,g\,\sin(\theta) &= m\,\frac{\partial^2 s}{\partial t^2} \\
-m\,g\,\sin(\theta) &= m\,l\,\frac{\partial^2 \theta}{\partial t^2} \\
\theta\text{"}(t) &= -\frac{g}{l}\sin(\theta(t))
\end{aligned}$$

For the linear pendulum problem, we assume that we have chosen a scale for our units so that $\frac{g}{l} = 1$, and we assume that θ is so small that we can take $\sin\theta \approx \theta$. Then our model system is:

$$\begin{aligned}
\theta\text{"}(t) &= -\theta(t) \\
\theta(t_0) &= u \\
\theta'(t_0) &= v \\
\theta_{exact}(t - t_0) &= u\cos(t) + v\sin(t)
\end{aligned}$$

For convenience, we might take $t_0 = 0$, $u = 1$, $v = 0$. Note that the linear pendulum model is a good approximation for small angles θ, so using $u = 1$ is stretching the model beyond the limits where it is a good approximation!

Here θ is the angle made with the vertical axis, measured in the counterclockwise direction, so that $\theta = 0$ corresponds to the pendulum hanging straight down, and a value $\theta = \frac{\pi}{2} \approx 1.5$ would have the pendulum pointing in the direction we might think of as 3 o'clock.

The quantity θ' measures the angular velocity, that is, the rate at which the angle theta is increasing. A positive value means the pendulum is traveling counterclockwise.

The initial conditions specify the angle and angular velocity at the initial time.

3 Euler method solution

As outlined in lst week's lab, we can use the Euler method to estimate the solution of a pendulum problem. First, of course, we transform the second order system into a pair of coupled first order equations. Then, using the default data, and specifing a fairly small number of steps n as an input quantity, we can compute with:

```
function pendulum_euler ( n )

  a = 0.0;
  b = 6.0 * pi;
  u = 1.0;
  v = 0.0;
  y0 = [ u, v ];

  [ t, y ] = euler ( @(t,y)pendulum_prime(t,y), n, a, b, y0 );

  return
end
```

with the derivative specified by:

```
function value = pendulum_prime ( t, y )

```

```
  u = y(1);
  v = y(2);

  dudt = v;
  dvdt = u;

  value = [ dudt; dvdt ];

  return
end
```

4 What happens to the solution as we change n?

Run the code *pendulum_euler(n)* using $n = 100$ steps. We get a solution, but are we comfortable with it? If we didn't know the exact solution, how could we judge our result?

Certainly, if our approximation is good, then using $n = 200$ steps over the same time interval should not change it appreciably. But when we plot the new solution, it seems noticeably different. Jumping ahead, we try $n = 1000$ steps and see that the approximate solution seems to reach roughly the same peak value at each period, rather than growing. This gives us some reason to trust the accuracy of the result.

Since we have the exact solution for this problem, we can run the code *pendulum_compare_euler(n)*, starting with $n = 100$, to see how the stepsize is affecting our accuracy. But in general, we won't know the solution of our problem, and so we need to rely on comparing the solution at two stepsizes to judge whether we are converging to a reasonable solution.

Another way to judge the solution arises because we happen to know that the solution ought to be periodic. This means that, instead of plotting the two solution components against t, we can consider plotting $\theta(t)$ versus $\theta'(t)$. This is known as a **phase plot**. A periodic solution will repeatedly trace the same closed curve. Let's go back to $n = 100$ and look at the corresponding phase plot. Now we can see that the trace of our approximate solution is not close to a closed curve. The gradual growth in size is obvious. **phase plot**

5 The pendulum with a friction term

The pendulum model can include a term for wind resistance or friction as a force that opposes the velocity, with a proportionality constant of α:

$$\begin{aligned} \theta"(t) &= -\alpha\theta'(t) - \theta(t) \\ \theta(t_0) &= u \\ \theta'(t_0) &= v \end{aligned}$$

Friction means that energy is lost, and that therefore, the true physical motion cannot be periodic. Instead, it must slowly decay towards zero. Does our computation also reveal this behavior?

The function `pendulum_friction_euler(n,alpha)` allows us to vary the number of steps n and the friction coefficient α in our code. To begin our exploration, we can use $n = 100$ and $\alpha = 0.05$.

While the time plots are useful, it is the phase plot that will be easiest to interpret.

n	alpha	Remarks
100	0.05	____________________
500	0.05	____________________
1000	0.05	____________________
2000	0.05	____________________

6 Controlling the pendulum with a friction term

For the linear pendulum with no friction, the exact solution is

$$\theta(t) = u\cos(t) + v\sin(t)$$
$$\theta'(t) = -u\sin(t) + v\cos(t)$$

from which we can conclude that something we can think of as the energy E satisfies:

$$E(t) = \theta(t)^2 + \theta'(t)^2 = u^2 + v^2 = \text{constant}$$

By contrast, we expect that energy of the friction pendulum will steadily decline towards zero from its initial value of $u^2 + v^2$. Could we monitor this decline, and give the pendulum a "kick" from time to time to boost its energy and keep it from running down? This is a simple version of a control problem.

To code it is simple. After each Euler step, we can compute the current energy $E(t)$. If it falls below, say, 90% of its initial value, we can simulate a jolt from an energy source that multiplies θ and θ' by a factor that restores the initial energy level.

We can use the function `pendulum_control_euler(n,alpha)` to do these experiments. This should make for an interesting phase plot!

7 The nonlinear pendulum model

A simple version of the nonlinear pendulum model has the form

$$\theta''(t) = -\sin(\theta(t))$$
$$\theta(t_0) = u$$
$$\theta'(t_0) = v$$

Again, we might take by default the values $t_0 = 0$, $u = 1$ and $v = 0$. We can run these experiments with `pendulum_nonlinear_euler(n)`.

For large values of θ, we might expect the linear and nonlinear to differ in some way, but we may have no idea of how this will happen.

But we can easily experiment, running both models and then comparing their plots, setting $v = 0$ and choosing a sequence of u values of increasing size. We can do this using the code `pendulum_angle_euler(n, theta0)`, where n is the number of time steps, and `theta0` is the initial value for the angular displacement . What can we observe?

8 Checking results against MATLAB's ode23()

The solution of differential equations is so important that, of course, MATLAB includes a number of ODE solvers. A simple one, which we will discuss in the next lab, is called `ode23()`. It has the advantage that it automatically estimates the appropriate stepsize to get good accuracy, so we don't need to worry about picking that ourselves.

The input is arranged a little differently than what we have used for our `euler()` code. In particular, the starting and stopping times are input as a vector. Here is an example of how we might solve the pendulum problem, as coded in `pendulum_ode23()`:

```
  [ t, y ] = ode23 ( @(t,y)pendulum_prime(t,y), [tstart, tstop], y0 ];
```

We may find it convenient to refer to `ode23()` to check the answers we get with our own ODE solvers.

9 No computing assignment for this lab!