Lesson 8: Ordinary Differential Equations and Euler's Method

1 Radioactive decay

Unstable nuclei like ^{235}U, can undergo radioactive decay and break apart into two smaller nuclei (along with other decay products). We cannot know precisely when a particular nucleus will decay due to the quantum mechanical uncertainty principle, but we can know the probability that it will have decayed after some time. In this sense, radioactive decay is a statistical problem.

Consider radioactive decay from another perspective. Rather than look at a single ^{235}U nucleus, look at a very large number of nuclei. If we know the probability that a single nucleus will decay after some period of time, then given a large number of nuclei, we can determine the fraction that remain after that period of time. Because the number of nuclei that will decay after some time is proportional to the number that were there to begin with, the rate of change of the number of nuclei is proportional to the number of nuclei. That is, $dN(t)/dt$ is proportional to $N(t)$, where $N(t)$ is the number of nuclei present at time t. Thus, the evolution of $N(t)$ is given by

$$\frac{dN(t)}{dt} = -\frac{N(t)}{\tau} \tag{1}$$

where the minus sign indicates that the population decreases in time. The constant τ is the mean lifetime (not the half–life) of the unstable nucleus.

We would like to develop a step–by–step procedure for solving this equation numerically, a procedure that can be programmed on a computer. Most differential equations cannot be solved analytically, so a numerical solution is the only option. In this simple example of radioactive decay, we are lucky: there does exist an exact, analytical solution. Let us consider the analytical solution first.

The exact solution is found by separation of variables. Writing $dN/N = -dt/\tau$ and integrating, we find

$$\ln N = -\frac{t}{\tau} + C$$

The constant C can be found by evaluating the result at $t = 0$. This gives $C = \ln N_0$, where N_0 is the number of nuclei at $t = 0$. The exact solution to the differential equation (1) for radioactive decay is then

$$N(t) = N_0 e^{-t/\tau} \tag{2}$$

As expected, the number of nuclei approaches zero exponentially. The half–life, obtained by setting $N(t) = N_0/2$, is $t_h = \tau \ln 2$.

2 Euler's method

Recall that the derivative is defined by

$$\frac{dN}{dt} = \lim_{\Delta t \to 0} \frac{\Delta N}{\Delta t}$$

That is, dN/dt is *approximately* equal to $\Delta N/\Delta t$ for small Δt. Replacing dN/dt with $\Delta N/\Delta t$ in Eq. (1), we find

$$\Delta N = -\frac{N}{\tau}\Delta t \tag{3}$$

We can solve this equation using the following step–by–step procedure. Let $N_0 \equiv N(0)$ denote the initial number of nuclei, at $t = 0$, and $N_1 \equiv N(\Delta t)$ denote the number of nuclei at time $t = \Delta t$. Evaluating Eq. (3) at $t = 0$ and identifying ΔN with $N_1 - N_0$, we have

$$N_1 = N_0 - \frac{N_0}{\tau}\Delta t \tag{4}$$

This gives us the number of nuclei at $t = \Delta t$. We can iterate this procedure. Evaluating Eq. (3) at $t = \Delta t$ and setting ΔN equal to $N_2 - N_1$, we have

$$N_2 = N_1 - \frac{N_1}{\tau}\Delta t \tag{5}$$

where N_2 is the number of nuclei at time $t = 2\Delta t$. In general, the number of nuclei at time $t = (i+1)\Delta t$ is given by

$$N_{i+1} = N_i - \frac{N_i}{\tau}\Delta t \tag{6}$$

where $N_i \equiv N(i\Delta t)$ is the number of nuclei at $t = i\Delta t$.

This step–by–step procedure is called *Euler's method*. It is the most simple, and least accurate, numerical method for solving a first order ordinary differential equation. Let's try it out.

> Exercise I: Start with an initial population of $N_0 = 100$ radioactive nuclei, and assume a mean lifetime of $\tau = 5\,\mathrm{s}$. Calculate by hand the number of nuclei left after $2\,\mathrm{s}$, using Eq. (6) with $\Delta t = 0.5\,\mathrm{s}$. Compare your answer to the exact result from the analytical solution (2). Repeat the calculation using $\Delta t = 0.25\,\mathrm{s}$. Is your new answer closer to the exact result?

Doing this calculation by hand is tiresome, especially if we want to use a small "timestep" Δt. Fortunately, computers don't get tired! Create a file with the following python code, which implements Euler's method for the solution of the radioactive decay problem:

```
import numpy as n

tau = 5.0        # mean lifetime
Ninitial = 100.0 # initial number of nuclei
tinitial = 0.0   # initial time
```

```
tfinal = 2.0     # final time
nts = 20         # number of timesteps
dt = (tfinal - tinitial)/nts

# Create arrays to hold times and numbers of nuclei
t = n.linspace(tinitial,tfinal,num=nts+1)
N = n.zeros(len(t))

# Evolve with Euler's method
N[0] = Ninitial
for i in range(0,nts):
    N[i+1] = N[i] - N[i]*dt/tau
    print 'at time t = {:.3f}, N = {:.3f}' .format(t[i+1],N[i+1])
```

In this code, the number of timesteps is specified by the variable `nts`, and the time evolution loop is controlled by a `for` statement. The timestep Δt, denoted `dt`, is computed as the total time divided by the number of timesteps.

Exercise II: Make sure you understand how the above code works. Run this code using various values of `nts`, and note that the answer improves as the resolution is increased. (We say that the resolution increases as the number of timesteps `nts` increases for a given time interval `tfinal - tinitial`. Equivalently, the resolution increases as the timestep Δt decreases.)

3 Discretization error

With a reasonable size value for Δt, the main source of error for Euler's method occurs when we approximate the derivative dN/dt at time $t = i\Delta t$ with the expression $(N_{i+1} - N_i)/\Delta t$. This type of error is called *discretization error* or *truncation error*.[1] We can understand the discretization error by considering the Taylor series expansion of $N(t + \Delta t)$ about time t:

$$N(t + \Delta t) = N(t) + \left(\frac{dN(t)}{dt}\right)\Delta t + \frac{1}{2}\left(\frac{d^2N(t)}{dt^2}\right)\Delta t^2 + \cdots \tag{7}$$

For notational simplicity, let $t_i \equiv i\Delta t$. Evaluating the Taylor series at $t = t_i$, and recognizing that $N(t_i) \equiv N_i$, we have

$$N_{i+1} = N_i + \left(\frac{dN(t)}{dt}\right)\Big|_{t_i}\Delta t + \frac{1}{2}\left(\frac{d^2N(t)}{dt^2}\right)\Big|_{t_i}\Delta t^2 + \cdots \tag{8}$$

Now use Eq. (1) to replace the first derivative of N. This yields

$$N_{i+1} = N_i - \frac{N_i}{\tau}\Delta t + \frac{1}{2}\left(\frac{d^2N(t)}{dt^2}\right)\Big|_{t_i}\Delta t^2 + \cdots \tag{9}$$

[1] For extremely small Δt, the numerical algorithm is also subject to machine roundoff error. This will be discussed in a later lesson.

Compare this with our Euler's method Eq. (6). We see that Euler's method is obtained by dropping the terms proportional to Δt^2, and higher order terms, in the exact expression (9).

With each timestep, when we use Euler's method to compute N_{i+1} from N_i, we introduce some error proportional to Δt^2. To reach the final time, $t = 2\,\mathrm{s}$ in the Exercise above, we must take `nts` timesteps. The total error that accumulates after `nts` timesteps is proportional to $\mathtt{nts}\Delta t^2$. Now note that the number of timesteps `nts` required to reach a given final time is proportional to $1/\Delta t$. Thus we find the key result that the error in Euler's method is proportional to Δt. If we double the resolution by doubling the number of timesteps and cutting Δt in half, the error is cut in half.

> Exercise III: Consider again the radioactive decay code. Using the exact result from the analytical solution in Eq. 2, find the error in Euler's method using $\mathtt{nts} = 5, 10, 20, 40$. Compute the ratio of successive errors. Is the error cut in half when the resolution is doubled?

Note that the error is not reduced by *exactly* one–half when `nts` is doubled. This is because the error in each timestep is not exactly proportional to Δt^2; rather, it is a series that also includes terms proportional to Δt^3, *etc.*

4 Free fall

Consider an object freely falling along the vertical y axis under the influence of gravity. The acceleration of the object is $-g$. Newton's second law tells us that the time rate of change of velocity is $dv/dt = -g$. The velocity is defined as the time rate of change of position, $v = dy/dt$. Thus, the motion of the object is governed by the two differential equations

$$\frac{dy}{dt} = v \tag{10a}$$

$$\frac{dv}{dt} = -g \tag{10b}$$

Of course, we can combine these two equations into the single second order differential equation $d^2y/dt^2 = -g$. But for numerical purposes, the first order form (10) is more useful.

To apply Euler's method, we replace the derivatives with finite differences and rearrange; this yields

$$y_{i+1} = y_i + v_i \Delta t \tag{11a}$$

$$v_{i+1} = v_i - g\Delta t \tag{11b}$$

Given the height y_i and velocity v_i at time $t_i \equiv i\Delta t$, these equations determine the height y_{i+1} and velocity v_{i+1} at time $t_{i+1} = t_i + \Delta t$.

Homework:

1. Write a python code that uses Euler's method to solve for the motion of an object in free fall. Choose the initial conditions $y_0 = 12.0\,\mathrm{m}$ and $v_0 = 35.0\,\mathrm{m/s}$ and set $g = 9.8\,\mathrm{m/s^2}$.

2. Use your code to determine the height for times $0 \leq t \leq 8\,\mathrm{s}$. Add the `pylab` module and have python plot a graph of y versus t. (Make sure your graph is properly labeled.)

3. Solve for the height of the object analytically, and (for the given initial conditions) find the exact height at $t = 8\,\mathrm{s}$.

4. Have your code compute the error in the height at $t = 8\,\mathrm{s}$. Do this for at least four different resolutions. Show that the error is approximately cut in half when the resolution is doubled.

5 Comment

In the code of section 2, we have specified directly the number of timesteps `nts`. The timestep itself is computed from `dt = (tfinal - tinitial)/nts`. This might seem surprising: why don't we specify the timestep `dt` directly? It is, in fact, quite common to do so. For example, we could replace the lines

```
nts = 20
dt = (tfinal - tinitial)/nts
t = n.linspace(tinitial,tfinal,num=nts+1)
```

with the following:

```
dt = 0.2
t = n.arange(tinitial,tfinal,dt)
```

For many purposes, this approach is perfectly acceptable. However, as discussed in Lesson 3, `arange` can be unpredictable when its arguments are real numbers. Depending on the values used, the final element of `t` can differ from `tfinal` by the amount `dt`. This might seem like a minor issue, but the difference of just one timestep in the final time can spoil the error analysis.

Another approach is to specify the timestep directly and compute the number of timesteps from `nts = (tfinal - tinitial)/dt`. Note, however, that this calculation yields a real number (float) for `nts`. If we want to use the `linspace` command, the number of timesteps must be an integer. We can force `nts` to be an integer by writing

```
dt = 0.2
nts = int((tfinal - tinitial)/dt)
t = n.linspace(tinitial,tfinal,num=nts+1)
```

This approach can potentially lead to problems as well. For example, let's say `tfinal - tinitial = 27.3` and `dt = 0.045`. Then `nts = 606`. Now double the resolution by dividing the timestep in half: `dt = 0.0225`. This gives `nts = 1213`. Although the timestep has been cut in half, the number of timesteps has *not* been doubled.