Physics 331 – Advanced Mechanics
Euler-Cromer Method

Not all of the differential equations encountered in this course (and elsewhere) can be solved analytically. However, there are numerical methods that can be used to solve them with a computer. You will have to solve about one problem per homework assignment computationally. We will focus on the solution of second order differential equations (i.e. involving first and second derivatives) because those are the most common in classical mechanics. For many problems, a fairly simple approach called the Euler-Cromer Method is sufficient. This method is easy to implement in Python or even lowly MS Excel.

Euler Method

The simplest numerical method to solve differential equations is the Euler Method. Suppose you want to find $x(t)$ and you know:

$$\frac{d^2x}{dt^2} = \ddot{x} = f\left(x, \frac{dx}{dt}, t\right)$$

Typically, you know the initial conditions of a system (i.e. values of x and $\dot{x} = \frac{dx}{dt}$ at some time). Appealing to the definition of the derivative, $\dot{x} = \lim_{\Delta t \to 0} \frac{x(t+\Delta t)}{\Delta t}$, if time is divided into merely *small* (instead of infinitesimal) steps Δt, then:

$$\dot{x} \approx \frac{x(t+\Delta t) - x(t)}{\Delta t} \qquad \text{and} \qquad \ddot{x} \approx \frac{\dot{x}(t+\Delta t) - \dot{x}(t)}{\Delta t} .$$

The first equation can be rearranged to approximate the solution a step forward in time:

$$x(t+\Delta t) \approx x(t) + \dot{x}\Delta t \qquad \text{or} \qquad x_{i+1} \approx x_i + \dot{x}_i \Delta t .$$

If x is a position coordinate, then we're essentially approximating that the speed remains constant over the time interval while updating the position.

Meanwhile, the second equation can be rearranged for updating the first derivative in terms of the second:

$$\dot{x}(t+\Delta t) \approx \dot{x}(t) + \ddot{x}\Delta t \qquad \text{or} \qquad \dot{x}_{i+1} \approx \dot{x}_i(t) + \ddot{x}_i \Delta t$$

We're essentially approximating that the acceleration (and corresponding forces) remains constant over the time interval while updating the velocity.

Now we have the new position, new velocity, and, since the acceleration can depend on these as well as on the time, we can calculate a new acceleration. Then we start all over using our newly calculated values to find *new* positions and velocities (and accelerations):

$$x_{i+2} \approx x_{i+1} + \dot{x}_{i+1}\Delta t , \quad \dot{x}_{i+2} \approx \dot{x}_{i+1}(t) + \ddot{x}_{i+1}\Delta t \ .$$

Plugging these two into each other is enlightening:

$$x_{i+2} \approx x_{i+1} + \left(\dot{x}_i(t) + \ddot{x}_i \Delta t\right)\Delta t = x_{i+1} + \dot{x}_i(t)\Delta t + \ddot{x}_i \left(\Delta t\right)^2$$

So, the new position depends on the current position and on the *previous* velocity and acceleration. Now, if there's a systematic evolution in the system's motion, then this

offset between which position and which velocity and acceleration are used can slowly add up to a big error.

Euler-Cromer Method

A simple modification to the Euler Method greatly improves the accuracy of the numerical solution. First, find the first derivative after a step forward in time as before:

$$\dot{x}(t+\Delta t) \approx \dot{x}(t) + \ddot{x}\Delta t \qquad \text{or} \qquad \dot{x}_{i+1} \approx \dot{x}_i(t) + \ddot{x}_i \Delta t$$

Next, use the derivative at this later time to step *x* forward in time with:

$$x(t+\Delta t) \approx x(t) + \underline{\dot{x}(t+\Delta t)}\Delta t \qquad \text{or} \qquad x_{i+1} \approx x_i + \underline{\dot{x}_{i+1}}\Delta t .$$

Note the slight difference between this and the Euler Method.

The derivative after two time steps is approximately:

$$\dot{x}_{i+2} \approx \dot{x}_{i+1}(t) + \ddot{x}_{i+1}\Delta t ,$$

and the solution after two steps is:

$$x_{i+2} \approx x_{i+1} + \underline{\dot{x}_{i+2}}\Delta t ,$$

and so on. Now, plugging these two into each other shows

$$x_{i+2} \approx x_{i+1} + \left(\dot{x}_{i+1}(t) + \ddot{x}_{i+1}\Delta t\right)\Delta t = x_{i+1} + \dot{x}_{i+1}(t)\Delta t + \ddot{x}_{i+1}\Delta t^2$$

By this method, the next position is calculated based on the current position, velocity, and time – the right-hand side is all in synch.

Implementation in Python

You may remember from Phys 231 writing lots of programs with lines like

Earth.pos = <xxx,xxx,xxx> $\qquad \vec{r}_o = \ldots$

Earth.p = <xxx.xxx.xxx> $\qquad \vec{p}_o = \ldots$

while t < tmax:

F=-G*Earth.m*Sun.m*Earth.pos/(mag(Earth.pos))**3 $\qquad \vec{F}_o = -G\dfrac{M_{sun}m_{Earth}}{r_o^{\,3}}\vec{r}_o$

Earth.p = Earth.p + F*deltat $\qquad \vec{p}_1 = \vec{p}_o + \vec{F}_o\Delta t$

Earth.pos = Earth.pos + (Earth.p/Earth.m)*deltat $\qquad \vec{r}_1 = \vec{r}_o + \dfrac{\vec{p}_1}{m}\Delta t$

t = t + deltat $\qquad t_1 = t_o + \Delta t$

Plugging the position and momentum equations into each other and rewriting in terms of position derivatives makes it evident that this is taking an Euler-Cromer approach:

$$\vec{r}_1 = \vec{r}_o + \frac{\vec{p}_o + \vec{F}_o \Delta t}{m}\Delta t = \vec{r}_o + \frac{\vec{p}_o}{m}\Delta t + \frac{\vec{F}_o}{m}(\Delta t)^2 = \vec{r}_o + \vec{v}_o \Delta t + \vec{a}_o (\Delta t)^2 = \vec{r}_o + \dot{\vec{r}}_o \Delta t + \ddot{\vec{r}}_o (\Delta t)^2$$

But be warned, simply flipping the three lines of code inside the loop would downgrade it to a mere Euler approach!

The code "EulerCromer VPython 1.py" shows how to implement the Euler-Cromer Method. "EulerCromer VPython F 1.py" explicitly applies it to model a ball's motion under the influence of a force. The code "EulerCromer Comparison.py" illustrates how much better this method works than the Euler Method. The solution to the differential equation in the example:

$$\frac{d^2x}{dt^2} = -kx,$$

should be a sinusoidal function. Notice that the amplitude of the solution with the Euler Method *increases* with time. However, the Euler-Cromer method gives a solution with a constant amplitude.

Modify EulerCromer VPython 1.py or EulerCromer VPython F 1.py to address the scenario at hand with the following steps:

1. Save a new copy have the original, and give the new copy an appropriate name.
2. Change the equation for a or F.
3. If necessary, change the initial conditions.
4. Change the time step, dt slightly to make sure that the solution does not change.

Practice Exercise

Modify the program to simulate a ball moving under the influence of a force like:

$$\ddot{x} = -4x - 0.2\dot{x}$$

(assuming appropriate units)

with the initial conditions $x(0) = 0$ and $\dot{x}(0) = 5$.