A Better Integrator

The results from Exercises 5.1 and 5.2 seem to be at odds with the discussion of errors just presented. You should have found that, for any choice of $\mathbf{a}$ for which an analytic solution is known, both the position and the velocity errors scale only as the *first* power of δt—in other words, reducing the timestep results in a far smaller improvement in accuracy than the earlier analysis would suggest. Why is this?

Part of the problem comes from the fact that the errors in our scheme are *coherent*, that is, errors in successive steps tend to be of the same sign—they don't jump around randomly. Thus, if the error after one step is ϵ, then the error after two is roughly 2ϵ, the error after N is $\sim N\epsilon$, and so on. (We assume here that the neglected terms in $\mathbf{a}'$ don't vary too much with time.) Since we are normally interested in integrating across a fixed time *interval*, the number of steps needed varies as $N \propto 1/\delta t$. We thus immediately lose one power of δt in our error assessments when we move from integration over a single step to integration over an interval.

A second problem is that the order of the $\mathbf{x}$ integration is greater than that of the $\mathbf{v}$ integration. Applying the reasoning of the previous paragraph, we come to the conclusion that the net scaling of the error in $\mathbf{v}$ is linear in δt, as observed. However, this in turn affects the $\mathbf{x}$ integration, since $\mathbf{v}$ appears in the second term. The result is that the per-step error in $\mathbf{x}$ acquires a term of the form $\delta\mathbf{v}\delta t$, i.e. $O(\delta t^2)$, and this becomes $O(\delta t)$ when integrated over a finite interval. The error in $\mathbf{v}$ actually *overwhelms* any benefit of including the $\frac{1}{2}\,\mathbf{a}\,\delta t^2$ term! You can verify this for yourself by simply omitting this term from your program.

What are we to do? One possibility is simply to accept the low order of the scheme and omit the extraneous acceleration term from the $\mathbf{x}$ equation. (For a clever reformulation of the problem that does just that, but nevertheless succeeds in retaining $O(\delta t^2)$ accuracy, read the section on the so-called **Leapfrog integrator**.) A better solution is to raise the order of the $\mathbf{v}$ integration by including a term in $\mathbf{j} \equiv d\mathbf{a}/dt$ (often referred to as the "jerk"). Sometimes we may know $\mathbf{a}'$ analytically, and we can include the "jerk" term explicitly. Often, however, that expression is too complex or too expensive to evaluate, and we must resort to other means.

The class of integration schemes collectively known as *predictor-corrector methods* in essence compute unknown higher derivatives by differencing lower-order terms, such as the acceleration. In our case, the procedure is to write

$$\mathbf{j}_i \;\approx\; \frac{\mathbf{a}_{i+1} - \mathbf{a}_i}{\delta t}\,,$$

and to augment the $\mathbf{v}$ equation to read

$$\mathbf{v}_{i+1} = \mathbf{v}_i + \mathbf{a}_i\,\delta t + \tfrac{1}{2}\,\mathbf{j}_i\,\delta t^2\,.$$

Methods such as this are called "predictor-corrector" because we first *predict* $\mathbf{x}$ and $\mathbf{v}$ using the acceleration at time t_i, then we evaluate the new acceleration at time t_{i+1} based on predicted position and velocity, and finally use that information to compute a *correction* to the velocity (and, in more general schemes, the position too). In the jargon of the field, this scheme is referred to as "predict-evaluate-correct," or "PEC." Many variants exist (for more information, read the appropriate section of *Numerical Recipes*).

A sample C++ code to implement this scheme (omitting all declarations and retaining only the integrator) is

```
// Prediction

a = acc(x, v, t);
x = x + v*dt + 0.5*a*dt*dt;
vp = v + a*dt;

// Correction

a_pred = acc(x, vp, t);
v = vp + 0.5*(a_pred-a)*dt;

t = t + dt;
```

Note that we could equally well combine the two "`v =`" lines to write

```
v = v + 0.5*(a+a_pred)*dt;
```

showing more clearly that we are simply evaluating the average acceleration over the interval (or, equivalently, evaluating the acceleration half a timestep ahead, as in the Leapfrog method). However, the first form above makes the PEC structure most evident.