1. Ordinary Differential Equations

Differential equations have the form:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f(x,t) \tag{1}$$

and can come both in systems (i.e. multiple equations that have to remain all true) and can contain higher orders in the derivatives. In the case above, we will want to solve for $x(t)$, and x is referred to as the dependent variable, and t is referred to as the independent variable.

What is the distinction between ordinary and partial differential equations?

Ordinary differential equations have a single independent variable, whereas *partial differential equations* require multiple independent variables.

For example, Poisson's equation:

$$\nabla^2\rho = 4\pi G\rho(x,y,z) \tag{2}$$

involves derivatives in all three coordinates, and they are all required in general. You can reduce Poisson's equation to an ODE in spherical symmetry.

What is the distinction between homogeneous and nonhomogeneous ODEs?

Homogeneous ODEs have no terms without the dependent variable. Nonhomogeneous ODEs have "driving" terms, without the dependent variable.

For example, a nonhomogeneous equation is:

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = f(t) \tag{3}$$

What is the distinction between linear and nonlinear ODEs?

Linear ODEs are *linear* in the dependent variable (but not necessarily in the independent variable).

Thus, the following equation is linear:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = t^3x(t) \tag{4}$$

but this equation is *nonlinear*:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = tx^3(t) \tag{5}$$

While solutions of homogeneous, linear ODEs may be superposed to find new solutions, the same is not true of nonlinear equations. Any two solutions to the top equation x_1 and x_2 can be

linearly combined into a new equation, but that's not going to be necessarily true of (nontrivial) solutions of the bottom equation.

What is meant by "first-order," "second-order," "third-order," etc. ODEs?

This nomenclature indicates the order of the derivatives involved. Thus, the following equation is first-order:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = w(x,t) \tag{6}$$

but this equation is second-order:

$$\frac{\mathrm{d}x}{\mathrm{d}t} + \alpha(x,t)\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = w(x,t) \tag{7}$$

With ODEs, you need to set *boundary conditions*. For example, in the simplest case, you might set a set of conditions at $t = 0$ (i.e. *initial conditions*). For each dependent variable, the number of independent boundary conditions you need to set are equal to the order of the problem.

You can in principle set conditions at multiple t, as long as you have the right number of conditions. In such cases, it is not necessarily the case that your conditions can be met under your equations of course. It definitely complicates the finding of a solution.

ODEs of any order can be reduced to systems of first-order equations. Specifically they can be reduced to a form like:

$$\begin{aligned} \frac{\mathrm{d}w_0}{\mathrm{d}t} &= f_0(\vec{w},t) \\ \frac{\mathrm{d}w_1}{\mathrm{d}t} &= f_1(\vec{w},t) \\ &\cdots \\ \frac{\mathrm{d}w_{N-1}}{\mathrm{d}t} &= f_{N-1}(\vec{w},t) \end{aligned} \tag{8}$$

where no terms appear on the right-hand side that are derivatives of $\vec{w}$ with t. In vector form this appears as:

$$\frac{\mathrm{d}\vec{w}}{\mathrm{d}t} = \vec{f}(\vec{w},t) \tag{9}$$

But do not confuse these vectors with three-dimensional vectors in configuration space. More often (usually) they involve vectors in phase space.

For example, take the second-order equation in three-dimensional space:

$$\frac{\mathrm{d}^2\vec{x}}{\mathrm{d}t^2} = -\vec{x} + \vec{f}(t) \tag{10}$$

which represents a spring under some time-dependent forcing. This may be reduced to one dimension as follows:

$$\frac{\mathrm{d}\vec{x}}{\mathrm{d}t} = \vec{v}$$

$$\frac{\mathrm{d}\vec{v}}{\mathrm{d}t} = -\vec{x} + \vec{f}(t) \tag{11}$$

which represents six first-order equations in the 6-D phase space vector $\vec{w} = (\vec{x}, \vec{v})$.

2. Algorithms for ODEs

For a general nonlinear ODE, given initial conditions, we will need to integrate (usually) forward in time numerically. There are a number of well-developed algorithms for doing this. As in other applications we have done this semester, the more complex algorithms tend to take the form of higher-order approximations to the solutions of the problem.

The simplest ODE integrator that exists is *Euler's algorithm*. Like all the simplest examples in this course, you should probably never use it, but it is illustrative.

This algorithm takes equal time steps Δt, starting from $\vec{w}(t = 0)$, just says:

$$\vec{w}(t + \Delta t) = \vec{w}(t) + \Delta t \dot{\vec{w}}(t) = \vec{w}(t) + \Delta t \vec{f}(\vec{w}, t) \tag{12}$$

What is the approximation error of this method for each step, and for a full integration, as a function of the step size?

For each step, the real answer can be derived from the Taylor Series:

$$\vec{w}(t + \Delta t) = \vec{w}(t) + \Delta t \dot{\vec{w}}(t) + \frac{\Delta t^2}{2} \ddot{\vec{w}}(t) + \ldots \tag{13}$$

So for each step the error is given by the third term, which is $\mathcal{O}(\Delta t^2)$. The total error is therefore, to leading order, the sum of the individual errors in the integral from $t = a$ to $t = b$:

$$E = \sum_{i=0}^{N-1} \frac{\Delta t^2}{2} \ddot{\vec{w}}(t_i) = \frac{\Delta t}{2} \sum_{i=0}^{N-1} \Delta t \dot{\vec{f}}(t_i) \approx \frac{\Delta t}{2} \int_a^b \mathrm{d}t \dot{\vec{f}}(t) = \frac{\Delta t}{2} \left[f(t = a) - f(t = b) \right]. \tag{14}$$

Therefore, a full integration's approximation error scales as $\mathcal{O}(\Delta t)$. In general, if an individual step is $\mathcal{O}(\Delta t^N)$ the full integration will be $\mathcal{O}(\Delta t^{N-1}$.

Second-order Runga-Kutta is simple second-order algorithm evaluates the function at $t + \Delta t/2$, and then takes the full step based on the values at the mid-point:

$$\begin{aligned} \vec{k}_1 &= \Delta t \vec{f}(\vec{w}_n, t_n) \\ \vec{k}_2 &= \Delta t \vec{f}\left(\vec{w}_n + \frac{1}{2} k_1, t_n + \frac{1}{2} \Delta t \right) \\ \vec{w}_{n+1} &= \vec{w}_n + \vec{k}_2 \end{aligned} \tag{15}$$

We can see why this is second-order fairly easily. In the case that we are integrating just one dependent variable, we are trying to approximate the integral:

$$\Delta w = \int_0^{\Delta t} \mathrm{d}t f(w(t), t) \tag{16}$$

We can Taylor expand $f(w(t),t)$ in t:

$$f(w,t) \approx f\left(\Delta t/2\right) + \left(t - \frac{\Delta t}{2}\right)\left[\frac{\mathrm{d}f}{\mathrm{d}t}\right]_{\Delta t/2} + \frac{1}{2}\left(t - \frac{\Delta t}{2}\right)^2\left[\frac{\mathrm{d}^2 f}{\mathrm{d}t^2}\right]_{\Delta t/2} + \ldots \tag{17}$$

When we insert this into the integral we find the second term vanishes:

$$\Delta w \approx \Delta t f(w(\Delta t/2), \Delta t/2) + \mathcal{O}(\Delta t^3) \tag{18}$$

This assumes that we have an estimate of $f(\Delta t/2)$. However, since this is multiplied by Δt, to remain second-order we only need this estimate to be good to first-order. Thus, the Euler algorithm can be used to estimate the midpoint and we get the procedure above.

This second-order Runge-Kutta algorithm requires two determinations of f, not one. So it will only be useful when the Δt^2 term is sufficiently small (i.e. the function is sufficiently smooth), so that the decrease in time step is worth it.

Harder to prove, but more useful generally, is the fourth-order Runge-Kutta. This is given by evaluating the derivative at the start, using that to estimate the derivative at the mid-point, then using the mid-point derivative to estimate the mid-point *again*, and and using those results to estimate a trial end point, before reaching the final answer.

$$\begin{aligned}
\vec{k}_1 &= \Delta t \vec{f}(\vec{w}_n, t_n) \\
\vec{k}_2 &= \Delta t \vec{f}(\vec{w}_n + \frac{1}{2}\vec{k}_1, t_n + \frac{1}{2}\Delta t) \\
\vec{k}_3 &= \Delta t \vec{f}(\vec{w}_n + \frac{1}{2}\vec{k}_2, t_n + \frac{1}{2}\Delta t) \\
\vec{k}_4 &= \Delta t \vec{f}(\vec{w}_n + \vec{k}_3, t_n + \Delta t) \\
\vec{w}_{n+1} &= \vec{w}_n + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4
\end{aligned} \tag{19}$$

This, it turns out, has error terms $\mathcal{O}(\Delta t^5)$.

3. Step sizes

In integrating differential equations, setting the step size is critical. Generally you don't know this in advance. ODE integrators essentially always use an adaptive step size designed to keep the integration accuracy within some tolerance.

Specifically, this is typically done by performing the integration in pairs of steps, and then comparing the answer using two steps to using one step twice as big. This adds at most 50% more function evaluations.

The error in one step can be written as $\Delta t^2 \phi$, where ϕ is some constant. The two answers will therefore be:

$$w(t + 2\Delta t) = w_1 + 2(\Delta t)^5 \phi + \mathcal{O}(\Delta t^6) \quad \text{two steps}$$

$$w(t+2\Delta t) \quad = \quad w_2 + (2\Delta t)^5\phi + \mathcal{O}(\Delta t^6) \quad \text{one step} \tag{20}$$

where ϕ is a constant of $\mathcal{O}(\Delta t^5)$. The difference gives you:

$$w_2 - w_1 \approx 30(\Delta t)^5\phi \tag{21}$$

The single-step absolute value error can therefore be estimated as:

$$\epsilon \approx \Delta t^5\phi \approx \frac{|w_2 - w_1|}{30} \tag{22}$$

Therefore, we can take the following approach to keep errors at some specified target level δ, in *per unit time* units. Per time step Δt, the corresponding error per step would be $\epsilon = \delta\Delta t$.

After each pair of steps, calculate ϵ. If $\epsilon < \Delta t\epsilon$, continue from $t + 2\Delta t$ (using w_2) but change the step size to a new one Δt_n going forward to satisfy:

$$\delta\Delta t_n = \Delta t_n^5\phi = \frac{\Delta t_n^5}{\Delta t^5}\Delta t^5\phi = \frac{\Delta t_n^5}{\Delta t^5}\frac{|w_2 - w_1|}{30} \tag{23}$$

which means:

$$\Delta t_n^4 = \Delta t^5\frac{30\delta}{|w_2 - w_1|} \tag{24}$$

and therefore:

$$\Delta t_n = \Delta t\left(\frac{30\delta\Delta t}{|w_2 - w_1|}\right)^{1/4} \tag{25}$$

It is wise to limit the increase in Δt (the right hand factor) to no more than a factor of two or so.

But if $\epsilon > \Delta t\epsilon$, then also set Δt_n using the same formula, but start over from t (that is, do not accept the initial measurement using the initial Δt.

There is a further improvement that can be made. We have an estimate of the one step error with step-size Δt to accuracy $\prime(\Delta t^6)$:

$$(\Delta t)^5\phi \approx \frac{w_2 - w_1}{30} \tag{26}$$

So you can correct the two-step estimate w_1:

$$w(t+2\Delta t) = w_1 + 2\Delta t^5\phi = w_1 + \frac{w_2 - w_1}{15} \tag{27}$$

for a fifth order integration method. This basically comes for free.

4. Leapfrog Method

Sometimes what we want is not just high order accuracy, but we want certain facts about the original equations to remain true, perhaps even at the expense of accuracy. For example, many

sets of equations are required to obey conservation laws, such as momentum, energy, etc. It is also sometimes nice to guarantee that the equations you are using are time-reversible, just like the actual equations in physics can be.

The simplest version of a time-reversible integrator is the *leapfrog method*. It is very similar to the second-order Runge-Kutta method. However, instead of:

$$\begin{aligned} \vec{k}_1 &= \Delta t \vec{f}(\vec{w}_n, t_n) \\ \vec{w}_{n+1/2} &= \vec{w}_n + \frac{1}{2}\vec{k}_1 \\ \vec{k}_2 &= \Delta t \vec{f}\left(\vec{w}_{n+1/2}, t_{n+1/2}\right) \\ \vec{w}_{n+1} &= \vec{w}_n + \vec{k}_2 \end{aligned} \tag{28}$$

We define:

$$\begin{aligned} \vec{k}_1 &= \Delta t \vec{f}(\vec{w}_n, t_n) \\ \vec{w}_{n+1/2} &= \vec{w}_{n-1/2} + \vec{k}_1 \\ \vec{k}_2 &= \Delta t \vec{f}(\vec{w}_{n+1/2}, t_{n+1/2}) \\ \vec{w}_{n+1} &= \vec{w}_n + \vec{k}_2 \end{aligned} \tag{29}$$

This means that if you start with some $\vec{w}_0$, you need to take a first half-step with Euler's method to $\vec{w}_{1/2}$, and then you can proceed from there. This is still second-order, but it can be shown that it is time-reversal-symmetric. This means that on average it will conserve energy. Believe it or not, this is not true of Runge-Kutta!

5. Verlet Method

In leapfrog, all components of $\vec{w}$ are updated at steps n and $n+1/2$. However, if there are a set of components of $\vec{w}$ whose update only depends on the intermediate values of the other components, and vice-versa, you can actually get away with only updating some of the components at n and the rest at $n + 1/2$. There is a useful and commonly used form of leapfrog that does exactly this, known as the *Verlet method*.

It is typically applied to classical equations of motion. If we write a one-dimensional equation of motion:

$$\begin{aligned} \dot{x} &= v \\ \dot{v} &= F(x) \end{aligned} \tag{30}$$

we can write:

$$\begin{aligned} x_1 &= x_0 + \Delta v_{1/2} \\ v_{3/2} &= v_{1/2} + \Delta F(x_1) \end{aligned} \tag{31}$$

This will be accurate with remainders of order Δ^3 at each time step. Note that eventually you want v_1, so:

$$v_1 = v_{1/2} + \Delta F(x_1)/2 \tag{32}$$

This method preserves time-reversibility. If I have $v_{3/2}$ and x_1, I can integrate backwards with the same equations to get the original $v_{1/2}$ and x_0. It also preserves phase space; a certain small volume in phase space would deform under the Verlet method in a way that conserved its volume. Another less generally useful fact, but still useful, is that in two or more dimensions in a spherically symmetric potential, this method conserves angular momentum. Preserving these facets of the physical system under study can be very useful.

6. Bulirsch-Stoer Method

The *Bulirsch-Stoer method* is a somewhat differently structured method. Here we will discuss it in one dimension, solving:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f(x,t) \tag{33}$$

Like other integrators, you proceed step by step. Here (as in the Newman book) we will call the step H from $x(t)$ to $x(t+H)$. In the end, we will be doing a long integration with many steps of size H. But within each one, we will be dividing it into intermediate steps; for each step H we will take 1 step of size $h_1 = H$, 2 steps of size $h_2 = H/2$, 3 steps of size $h_3 = H/3$, 4 steps of size $h_4 = H/4$, ... up to about 8 steps. Each will give an answer for the full integration across step H, and we will the series to extrapolate to a more exact answer.

The Bulirsch-Stoer works on the basis that the integration method for the intermediate steps have only even-order error terms. That is, that the leading order error will scale as $\mathcal{O}(h_n^2)$, and that if that is cancelled, the next leading order will be $\mathcal{O}(h_n^4)$, etc. If you have some number of steps n, then you can denote as $R_{n,1}$ the $\mathcal{O}(h_n^2)$ estimate. You can denote further as $R_{n,m}$ estimates with errors $\mathcal{O}(h_n^{2m})$.

$R_{n,m}$ will relate to the real answer as:

$$x(t+H) = R_{n,m} + c_m h_n^{2m} + \mathcal{O}(h_n^{2m+2}). \tag{34}$$

If you have also calculated the answer $R_{n-1,m}$ with one less step, you also have:

$$x(t+H) = R_{n-1,m} + c_m h_{n-1}^{2m} + \mathcal{O}(h_{n-1}^{2m+2}). \tag{35}$$

Because you know $h_{n-1} = nh_n/(n-1)$, you can determine c_m. If you know c_m, then you can construct a $2m+2$-order estimate:

$$R_{n,m+1} = R_{n,m} + c_m h_n^{2m}, \tag{36}$$

and in detail you can show this to be:

$$R_{n,m+1} = R_{n,m} + \frac{R_{n,m} - R_{n-1,m}}{[n/(n-1)]^{2m} - 1} \tag{37}$$

This equation allows us to build successively higher-order estimates. One starts with the one-step estimate $R_{1,1}$ of order $\mathcal{O}(h_1^2)$. Then you make the two-step estimate $R_{2,1}$ of order $\mathcal{O}(h_2^2)$. These first two estimates allow you to construct the two-step estimate of order $\mathcal{O}(h_2^4)$ from the recursion. Then you can similar use the three-step estimate $R_{3,1}$ and use the previous results and the recursion relation to estimate $R_{3,3}$, which is $\mathcal{O}(h_3^6)$. This is getting better and better!

The error estimate at each step is $c_m h_n^{2m}$. You keep increasing n until the error on $R_{n,n}$ is less than the target value, or you reach some limit, usually $n = 8$. If you reach the limit, you make H smaller, in a manner similar to that used in the adaptive step size approach described above.

Now we just need to find a way to take the intermediate steps which has an error which is an even function of the step size, so it is only the terms h^{2m} that need concern us. Notice that in the argument above we were depending on the errors to have that property. This will be true for any time-reversal-symmetric method. This is because such methods will have the property that their error per step must have the property:

$$\epsilon(-h) = -\epsilon(h) \tag{38}$$

So any Taylor expansion of the error per step has only odd terms. Since the full error of an integration will scale as one order higher, the full error has only even terms.

Leapfrog has this property, except for at the outset of the integration, where you need to do one half-step of Euler. That makes it obviously unsuited to the Bulirsch-Stoer method, which is taking a small number of steps h within H, and the error in that half-step will matter a lot and introduce undesirable odd-order terms.

Bulirsch-Stoer is designed to use the *modified midpoint method* instead, which does not have this problem. Starting at step 0:

$$\begin{aligned}
\vec{w}_{1/2} &= \vec{w}_0 + \frac{1}{2} h \vec{f}(\vec{w}_0, t_0) \\
\vec{w}_1 &= \vec{w}_0 + h \vec{f}(\vec{w}_{1/2}, t_{1/2}) \\
\vec{w}_{3/2} &= \vec{w}_{1/2} + h \vec{f}(\vec{w}_1, t_1) \\
\vec{w}_2 &= \vec{w}_1 + h \vec{f}(\vec{w}_{3/2}, t_{3/2}) \\
\ldots \quad (39)
\end{aligned}$$

There are two final points $w_{n-1/2}$ and w_n. We can also extrapolate $w_{n-1/2}$ to a separate estimate of w_n as follows:

$$w'_n = w_{n-1/2} + \frac{1}{2} h \vec{f}(\vec{w}_n, t_n) \tag{40}$$

We can average the two:

$$w_{n,\text{final}} = \frac{1}{2}\left(w_n + w'_n\right). \tag{41}$$

It can be shown that this estimate has the desired property that it only has even powers of h, by using the last step to cancel the odd powers of h introduced in the first step.