Chapter 13

Ordinary Differential Equations

We motivated the problem of interpolation in Chapter 11 by transitioning from *analzying* to *finding* functions. That is, in problems like interpolation and regression, the unknown is a function f, and the job of the algorithm is to fill in missing data.

We continue this discussion by considering similar problems involving filling in function values. Here, our unknown continues to be a function f, but rather than simply guessing missing values we would like to solve more complex design problems. For example, consider the following problems:

- Find f approximating some other function f_0 but satisfying additional criteria (smoothness, continuity, low-frequency, etc.).
- Simulate some dynamical or physical relationship as $f(t)$ where t is time.
- Find f with similar values to f_0 but certain properties in common with a different function g_0.

In each of these cases, our unknown is a function f, but our criteria for success is more involved than "matches a given set of data points."

The theories of ordinary differential equations (ODEs) and partial differential equations (PDEs) study the case where we wish to find a function $f(\vec{x})$ based on information about or relationships between its derivatives. Notice we already have solved a simple version of this problem in our discussion of quadrature: Given $f'(t)$, methods for quadrature provide ways of approximating $f(t)$ using integration.

In this chapter, we will consider the case of an *ordinary* differential equations and in particular *initial value problems*. Here, the unknown is a function $f(t) : \mathbb{R} \to \mathbb{R}^n$, and we given an equation satisfied by f and its derivatives as well as $f(0)$; our goal is to predict $f(t)$ for $t > 0$. We will provide several examples of ODEs appearing in the computer science literature and then will proceed to describe common solution techniques.

As a note, we will use the notation f' to denote the derivative df/dt of $f : [0, \infty) \to \mathbb{R}^n$. Our goal will be to find $f(t)$ given relationships between t, $f(t)$, $f'(t)$, $f''(t)$, and so on.

13.1 Motivation

ODEs appear in nearly any part of scientific example, and it is not difficult to encounter practical situations requiring their solution. For instance, the basic laws of physical motion are given by an ODE:

Example 13.1 (Newton's Second Law)**.** *Continuing from §5.1.2, recall that Newton's Second Law of Motion states that $F = ma$, that is, the total force on an object is equal to its mass times its acceleration. If we simulate n particles simultaneously, then we can think of combining all their positions into a vector $\vec{x} \in \mathbb{R}^{3n}$. Similarly, we can write a function $\vec{F}(t, \vec{x}, \vec{x}') \in \mathbb{R}^{3n}$ taking the time, positions of the particles, and their velocities and returning the total force on each particle. This function can take into account interrelationships between particles (e.g. gravitational forces or springs), external effects like wind resistance (which depends on $\vec{x}'$), external forces varying with time t, and so on.*

Then, to find the positions of all the particles as functions of time, we wish to solve the equation $\vec{x}'' = \vec{F}(t, \vec{x}, \vec{x}')/m$. We usually are given the positions and velocities of all the particles at time $t = 0$ as a starting condition.

Example 13.2 (Protein folding)**.** *On a smaller scale, the equations governing motions of molecules also are ordinary differential equations. One particularly challenging case is that of* protein folding*, in which the geometry structure of a protein is predicted by simulating intermolecular forces over time. These forces take many often nonlinear forms that continue to challenge researchers in computational biology.*

Example 13.3 (Gradient descent)**.** *Suppose we are wishing to minimize an energy function $E(\vec{x})$ over all $\vec{x}$. We learned in Chapter 8 that $-\nabla E(\vec{x})$ points in the direction E decreases the most at a given $\vec{x}$, so we did* line search *along that direction from $\vec{x}$ to minimize E locally. An alternative option popular in certain theories is to solve an ODE of the form $\vec{x}' = -\nabla E(\vec{x})$; in other words, think of $\vec{x}$ as a function of time $\vec{x}(t)$ that is attempting to decrease E by walking downhill.*

For example, suppose we wish to solve $A\vec{x} = \vec{b}$ for symmetric positive definite A. We know from §10.1.1 that this is equivalent to minimizing $E(\vec{x}) \equiv \frac{1}{2}\vec{x}^\top A\vec{x} - \vec{b}^\top\vec{x} + c$. Thus, we could attempt to solve the ODE $\vec{x}' = -\nabla f(\vec{x}) = \vec{b} - A\vec{x}$. As $t \to \infty$, we expect $\vec{x}(t)$ to better and better satisfy the linear system.

Example 13.4 (Crowd simulation)**.** *Suppose we are writing video game software requiring realistic simulation of virtual crowds of humans, animals, spaceships, and the like. One strategy for generating plausible motion, illustrated in Figure NUMBER, is to use differential equations. Here, the velocity of a member of the crowd is determined as a function of its environment; for example, in human crowds the proximity of other humans, distance to obstacles, and so on can affect the direction a given agent is moving. These rules can be simple, but in the aggregate their interaction is complex. Stable integrators for differential equations underlie this machinery, since we do not wish to have noticeably unrealistic or unphysical behavior.*

13.2 Theory of ODEs

A full treatment of the theory of ordinary differential equations is outside the scope of our discussion, and we refer the reader to CITE for more details. This aside, we mention some highlights here that will be relevant to our development in future sections.

13.2.1 Basic Notions

The most general ODE initial value problem takes the following form:

$$\begin{aligned}&\text{Find } f(t) : \mathbb{R} \to \mathbb{R}^n\\ &\text{Satisfying } F[t, f(t), f'(t), f''(t), \dots, f^{(k)}(t)] = 0\\ &\text{Given } f(0), f'(0), f''(0), \dots, f^{(k-1)}(0)\end{aligned}$$

Here, F is some relationship between f and all its derivatives; we use $f^{(\ell)}$ to denote the ℓ-th derivative of f. We can think of ODEs as determining *evolution* of f over time t; we know f and its derivatives at time zero and wish to predict it moving forward.

ODEs take many forms even in a single variable. For instance, denote $y = f(t)$ and suppose $y \in \mathbb{R}^1$. Then, examples of ODEs include:

- $y' = 1 + \cos t$: This ODE can be solved by integrating both sides e.g. using quadrature methods
- $y' = ay$: This ODE is linear in y
- $y' = ay + e^t$: This ODE is time and position-dependent
- $y'' + 3y' - y = t$: This ODE involves multiple derivatives of y
- $y'' \sin y = e^{ty'}$: This ODE is nonlinear in y and t.

Obviously the most general ODEs can be challenging to solve. We will restrict most of our discussion to the case of *explicit* ODEs, in which the highest-order derivative can be isolated:

Definition 13.1 (Explicit ODE)**.** *An ODE is* explicit *if can be written in the form*

$$f^{(k)}(t) = F[t, f(t), f'(t), f''(t), \dots, f^{(k-1)}(t)].$$

For example, an explicit form of Newton's second law is $\vec{x}''(t) = \frac{1}{m}\vec{a}(t, \vec{x}(t), \vec{x}'(t))$.

Surprisingly, generalizing the trick in §5.1.2, in fact any explicit ODE can be converted to a first-order equation $f'(t) = F[t, f(t)]$, where f has multidimensional output. This observation implies that we need no more than one derivative in our treatment of ODE algorithms. To see this relationship, we simply recall that $d^2y/dt^2 = d/dt(dy/dt)$. Thus, we can define an intermediate variable $z \equiv dy/dt$, and understand d^2y/dt^2 as dz/dt with the constraint $z = dy/dt$. More generally, if we wish to solve the explicit problem

$$f^{(k)}(t) = F[t, f(t), f'(t), f''(t), \dots, f^{(k-1)}(t)],$$

where $f : \mathbb{R} \to \mathbb{R}^n$, then we define $g(t) : \mathbb{R} \to \mathbb{R}^{kn}$ using the first-order ODE:

$$\frac{d}{dt}\begin{pmatrix} g_1(t) \\ g_2(t) \\ \vdots \\ g_{k-1}(t) \\ g_k(t) \end{pmatrix} = \begin{pmatrix} g_2(t) \\ g_3(t) \\ \vdots \\ g_k(t) \\ F[t, g_1(t), g_2(t), \dots, g_{k-1}(t)] \end{pmatrix}$$

Here, we denote $g_i(t) : \mathbb{R} \to \mathbb{R}^n$ to contain n components of g. Then, $g_1(t)$ satisfies the original ODE. To see so, we just check that our equation above implies $g_2(t) = g_1'(t)$, $g_3(t) = g_2'(t) = g_1''(t)$, and so on. Thus, making these substitutions shows that the final row encodes the original ODE.

The trick above will simplify our notation, but some care should be taken to understand that this approach does not trivialize computations. In particular, in may cases our function $f(t)$ will only have a single output, but the ODE will be in several derivatives. We replace this case with one derivative and several outputs.

Example 13.5 (ODE expansion)**.** *Suppose we wish to solve $y''' = 3y'' - 2y' + y$ where $y(t) : \mathbb{R} \to \mathbb{R}$. This equation is equivalent to:*

$$\frac{d}{dt}\begin{pmatrix} y \\ z \\ w \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & -2 & 3 \end{pmatrix}\begin{pmatrix} y \\ z \\ w \end{pmatrix}$$

Just as our trick above allows us to consider only first-order ODEs, we can restrict our notation even more to *autonomous* ODEs. These equations are of the form $f'(t) = F[f(t)]$, that is, F no longer depends on t. To do so, we could define

$$g(t) \equiv \begin{pmatrix} f(t) \\ \bar{g}(t) \end{pmatrix}.$$

Then, we can solve the following ODE for g instead:

$$g'(t) = \begin{pmatrix} f'(t) \\ \bar{g}'(t) \end{pmatrix} = \begin{pmatrix} F[f(t), \bar{g}(t)] \\ 1 \end{pmatrix}.$$

In particular, $\bar{g}(t) = t$ assuming we take $\bar{g}(0) = 0$.

It is possible to visualize the behavior of ODEs in many ways, illustrated in Figure NUMBER. For instance, if the unknown $f(t)$ is a function of a single variable, then we can think of $F[f(t)]$ as providing the slope of $f(t)$, as shown in Figure NUMBER. Alternatively, if $f(t)$ has output in $\mathbb{R}^2$, we no longer can visualize the dependence on time t, but we can draw *phase space*, which shows the tangent of $f(t)$ at each $(x, y) \in \mathbb{R}^2$.

13.2.2 Existence and Uniqueness

Before we can proceed to discretizations of the initial value problem, we should briefly acknowledge that not all differential equation problems are solvable. Furthermore, some differential equations admit multiple solutions.

Example 13.6 (Unsolvable ODE)**.** *Consider the equation $y' = 2y/t$, with $y(0) \neq 0$ given; notice we are not dividing by zero because $y(0)$ is prescribed. Rewriting as*

$$\frac{1}{y}\frac{dy}{dt} = \frac{2}{t}$$

and integrating with respect to t on both sides shows:

$$\ln|y| = 2\ln t + c$$

Or equivalently, $y = Ct^2$ for some $C \in \mathbb{R}$. Notice that $y(0) = 0$ in this expression, contradicting our initial conditions. Thus, this ODE has no solution with the given initial conditions.

Example 13.7 (Nonunique solutions)**.** *Now, consider the same ODE with $y(0) = 0$. Consider $y(t)$ given by $y(t) = Ct^2$ for* any *$C \in \mathbb{R}$. Then, $y'(t) = 2Ct$. Thus,*

$$\frac{2y}{t} = \frac{2Ct^2}{t} = 2Ct = y'(t),$$

showing that the ODE is solved by this function regardless of C. Thus, solutions of this problem are nonunique.

Thankfully, there is a rich theory characterizing behavior and stability of solutions to differential equations. Our development in the next chapter will have a stronger set of conditions needed for existence of a solution, but in fact under weak conditions on f it is possible to show that an ODE $f'(t) = F[f(t)]$ has a solution. For instance, one such theorem guarantees local existence of a solution:

Theorem 13.1 (Local existence and uniqueness)**.** *Suppose F is continuous and Lipschitz, that is, $\|F[\vec{y}] - F[\vec{x}]\|_2 \leq L\|\vec{y} - \vec{x}\|_2$ for some L. Then, the ODE $f'(t) = F[f(t)]$ admits exactly one solution for all $t \geq 0$ regardless of initial conditions.*

In our subsequent development, we will assume that the ODE we are attempting to solve satisfies the conditions of such a theorem; this assumption is fairly realistic in that at least locally there would have to be fairly degenerate behavior to break such weak assumptions.

13.2.3 Model Equations

One way to gain intuition for the behavior of ODEs is to examine behavior of solutions to some simple *model equations* that can be solved in closed form. These equations represent linearizations of more practical equations, and thus locally they model the type of behavior we can expect.

We start with ODEs in a single variable. Given our simplifications in §13.2.1, the simplest equation we might expect to work with would be $y' = F[y]$, where $y(t) : \mathbb{R} \to \mathbb{R}$. Taking a linear approximation would yield equations of type $y' = ay + b$. Substituting $\bar{y} \equiv y + {}^{b}/_{a}$ shows: $\bar{y}' = y' = ay + b = a(\bar{y} - {}^{b}/_{a}) + b = a\bar{y}$. Thus, in our model equations the constant b simply induces a shift, and for our phenomenological study in this section we can assume $b = 0$.

By the argument above, we locally can understand behavior of $y' = F[y]$ by studying the linear equation $y' = ay$. In fact, applying standard arguments from calculus shows that

$$y(t) = Ce^{at}.$$

Obviously, there are three cases, illustrated in Figure NUMBER:

1. $a > 0$: In this case solutions get larger and larger; in fact, if $y(t)$ and $\hat{y}(t)$ both satisfy the ODE with slightly different starting conditions, as $t \to \infty$ they diverge.

2. $a = 0$: The system in this case is solved by constants; solutions with different starting points stay the same distance apart.

3. $a < 0$: Then, all solutions of the ODE approach 0 as $t \to \infty$.

We say cases 2 and 3 are *stable*, in the sense that perturbing $y(0)$ slightly yields solutions that get close and close over time; case 1 is *unstable*, since a small mistake in specifying the input parameter $y(0)$ will be amplified as time t advances. Unstable ODEs generate ill-posed computational problems; without careful consideration we cannot expect numerical methods to generate usable solutions in this case, since even theoretical outputs are so sensitive to perturbations of the input. On the other hand, stable problems are well-posed since small mistakes in $y(0)$ get diminished over time.

Advancing to multiple dimensions, we could study the linearized equation

$$\vec{y}' = A\vec{y}.$$

As explained in §5.1.2, if $\vec{y}_1, \cdots, \vec{y}_k$ are eigenvectors of A with eigenvalues $\lambda_1, \ldots, \lambda_k$ and $\vec{y}(0) = c_1\vec{y}_1 + \cdots + c_k\vec{y}_k$, then

$$\vec{y}(t) = c_1 e^{\lambda_1 t}\vec{y}_1 + \cdots + c_k e^{\lambda_k t}\vec{y}_k.$$

In other words, the eigenvalues of A take the place of a in our one-dimensional example. From this result, it is not hard to intuit that a multivariable system is stable exactly when its spectral radius is less than one.

In reality we wish to solve $\vec{y}' = F[\vec{y}]$ for general functions F. Assuming F is differentiable, we can write $F[\vec{y}] \approx F[\vec{y}_0] + J_F(\vec{y}_0)(\vec{y} - \vec{y}_0)$, yielding the model equation above after a shift. Thus, for short periods of time we expect behavior similar to the model equation. Additionally, the conditions in Theorem 13.1 can be viewed as a bound on the behavior of J_F, providing a connection to less localized theories of ODE.

13.3 Time-Stepping Schemes

We now proceed to describe several methods for solving the nonlinear ODE $\vec{y}' = F[\vec{y}]$ for potentially nonlinear functions F. In general, given a "time step" h, our methods will be used to generate estimates of $\vec{y}(t+h)$ given $\vec{y}(t)$. Applying these methods iteratively generates estimates of $\vec{y}_0 \equiv \vec{y}(t)$, $\vec{y}_1 \equiv \vec{y}(t+h)$, $\vec{y}_2 \equiv \vec{y}(t+2h)$, $\vec{y}_3 \equiv \vec{y}(t+3h)$, and so on. Notice that since F has no t dependence the mechanism for generating each additional step is the same as the first, so for the most part we will only need to describe a single step of these methods. We call methods for generating approximations of $\vec{y}(t)$ *integrators*, reflecting the fact that they are integrating out the derivatives in the input equation.

Of key importance to our consideration is the idea of *stability*. Just as ODEs can be stable or unstable, so can discretizations. For example, if h is too large, some schemes will accumulate error at an exponential rate; contrastingly, other methods are stable in that even if h is large the solutions will remain bounded. Stability, however, can compete with *accuracy*; often time stable schemes are bad approximations of $\vec{y}(t)$, even if they are guaranteed not to have wild behavior.

13.3.1 Forward Euler

Our first ODE strategy comes from our construction of the forward differencing scheme in §12.3.2:

$$F[\vec{y}_k] = \vec{y}'(t) = \frac{\vec{y}_{k+1} - \vec{y}_k}{h} + O(h)$$

Solving this relationship for $\vec{y}_{k+1}$ shows

$$\vec{y}_{k+1} = \vec{y}_k + hF[\vec{y}_k] + O(h^2) \approx \vec{y}_k + hF[\vec{y}_k].$$

Thus, the *forward Euler* scheme applies the formula on the right to estimate $\vec{y}_{k+1}$. It is one of the most efficient strategies for time-stepping, since it simply evaluates F and adds a multiple of the result to $\vec{y}_k$. For this reason, we call it an *explicit* method, that is, there is an explicit formula for $\vec{y}_{k+1}$ in terms of $\vec{y}_k$ and F.

Analyzing the accuracy of this method is fairly straightforward. Notice that our approximation of $\vec{y}_{k+1}$ is $O(h^2)$, so each step induces quadratic error. We call this error *localized truncation error* because it is the error induced by a single step; the word "truncation" refers to the fact that we truncated a Taylor series to obtain this formula. Of course, our iterate $\vec{y}_k$ already may be inaccurate thanks to accumulated truncation errors from previous iterations. If we integrate from t_0 to t with $O(1/h)$ steps, then our total error looks like $O(h)$; this estimate represents *global truncation error*, and thus we usually write that the forward Euler scheme is "first-order accurate."

The stability of this method requires somewhat more consideration. In our discussion, we will work out the stability of methods in the one-variable case $y' = ay$, with the intuition that similar statements carry over to multidimensional equations by replacing a with the spectral radius. In this case, we know

$$y_{k+1} = y_k + ahy_k = (1 + ah)y_k.$$

In other words, $y_k = (1 + ah)^k y_0$. Thus, the integrator is stable when $|1 + ah| \leq 1$, since otherwise $|y_k| \to \infty$ exponentially. Assuming $a < 0$ (otherwise the problem is ill-posed), we can simplify:

$$\begin{aligned} |1 + ah| \leq 1 &\iff -1 \leq 1 + ah \leq 1 \\ &\iff -2 \leq ah \leq 0 \\ &\iff 0 \leq h \leq \frac{2}{|a|}, \text{ since } a < 0 \end{aligned}$$

Thus, forward Euler admits a *time step restriction* for stability given by our final condition on h. In other words, the output of forward Euler can explode even when $y' = ay$ is stable if h is not small enough. Figure NUMBER illustrates what happens when this condition is obeyed or violated. In multiple dimensions, we can replace this restriction with an analogous one using the spectral radius of A. For nonlinear ODEs this formula gives a guide for stability at least locally in time; globally h may have to be adjusted if the Jacobian of F becomes worse conditioned.

13.3.2 Backward Euler

Similarly, we could have applied the *backward* differencing scheme at $\vec{y}_{k+1}$ to design an ODE integrator:

$$F[\vec{y}_{k+1}] = \vec{y}'(t) = \frac{\vec{y}_{k+1} - \vec{y}_k}{h} + O(h)$$

Thus, we solve the following potentially nonlinear system of equations for $\vec{y}_{k+1}$:

$$\vec{y}_k = \vec{y}_{k+1} - hF[\vec{y}_{k+1}].$$

Because we have to solve this equation for $\vec{y}_{k+1}$, backward Euler is an *implicit* integrator.

This method is first-order accurate like forward Euler by an identical proof. The stability of this method, however, contrasts considerably with forward Euler. Once again considering the model equation $y' = ay$, we write:

$$y_k = y_{k+1} - hay_{k+1} \implies y_{k+1} = \frac{y_k}{1-ha}.$$

Paralleling our previous argument, backward Euler is stable under the following condition:

$$\begin{aligned}\frac{1}{|1-ha|} \leq 1 &\iff |1-ha| \geq 1 \\ &\iff 1-ha \leq -1 \text{ or } 1-ha \geq 1 \\ &\iff h \leq \frac{2}{a} \text{ or } h \geq 0, \text{ for } a < 0\end{aligned}$$

Obviously we always take $h \geq 0$, so backward Euler is unconditionally stable.

Of course, even if backward Euler is stable it is not necessarily accurate. If h is too large, $\vec{y}_k$ will approach zero far too quickly. When simulating cloth and other physical materials that require lots of high-frequency detail to be realistic, backward Euler may not be an effective choice. Furthermore, we have to invert $F[\cdot]$ to solve for $\vec{y}_{k+1}$.

Example 13.8 (Backward Euler)**.** *Suppose we wish to solve $\vec{y}' = A\vec{y}$ for $A \in \mathbb{R}^{n\times n}$. Then, to find $\vec{y}_{k+1}$ we solve the following system:*

$$\vec{y}_k = \vec{y}_{k+1} - hA\vec{y}_{k+1} \implies \vec{y}_{k+1} = (I_{n\times n} - hA)^{-1}\vec{y}_k.$$

13.3.3 Trapezoidal Method

Suppose $\vec{y}_k$ is known at time t_k and $\vec{y}_{k+1}$ represents the value at time $t_{k+1} = t_k + h$. Suppose we also know $\vec{y}_{k+1/2}$ halfway in between these two steps. Then, by our derivation of the centered differencing we know:

$$\vec{y}_{k+1} = \vec{y}_k + hF[\vec{y}_{k+1/2}] + O(h^3)$$

From our derivation of the trapezoidal rule:

$$\frac{F[\vec{y}_{k+1}] + F[\vec{y}_k]}{2} = F[\vec{y}_{k+1/2}] + O(h^2)$$

Substituting this relationship yields our first second-order integration scheme, the *trapezoid method* for integrating ODEs:

$$\vec{y}_{k+1} = \vec{y}_k + h\frac{F[\vec{y}_{k+1}] + F[\vec{y}_k]}{2}$$

Like backward Euler, this method is implicit since we must solve this equation for $\vec{y}_{k+1}$.

Once again carrying out stability analysis on $y' = ay$, we find in this case time steps of the trapezoidal method solve

$$y_{k+1} = y_k + \frac{1}{2}ha(y_{k+1} + y_k)$$

In other words,

$$y_k = \left(\frac{1+\frac{1}{2}ha}{1-\frac{1}{2}ha}\right)^k y_0.$$

The method is thus stable when

$$\left|\frac{1+\frac{1}{2}ha}{1-\frac{1}{2}ha}\right| < 1.$$

It is easy to see that this inequality holds whenever $a < 0$ and $h > 0$, showing that the trapezoid method is unconditionally stable.

Despite its higher order of accuracy with maintained stability, the trapezoid method, however, has some drawbacks that make it less popular than backward Euler. In particular, consider the ratio

$$R \equiv \frac{y_{k+1}}{y_k} = \frac{1+\frac{1}{2}ha}{1-\frac{1}{2}ha}$$

When $a < 0$, for large enough h this ratio eventually becomes negative; in fact, as $h \to \infty$, we have $R \to -1$. Thus, as illustrated in Figure NUMBER, if time steps are too large, the trapezoidal method of integration tends to exhibit undesirable oscillatory behavior that is not at all like what we might expect for solutions of $y' = ay$.

13.3.4 Runge-Kutta Methods

A class of integrators can be derived by making the following observation:

$$\begin{aligned}\vec{y}_{k+1} &= \vec{y}_k + \int_{t_k}^{t_k+\Delta t} \vec{y}'(t)\,dt \text{ by the Fundamental Theorem of Calculus} \\ &= \vec{y}_k + \int_{t_k}^{t_k+\Delta t} F[\vec{y}(t)]\,dt\end{aligned}$$

Of course, using this formula outright does not work for formulating a method for time-stepping since we do not know $\vec{y}(t)$, but careful application of our quadrature formulae from the previous chapter can generate feasible strategies.

For example, suppose we apply the trapezoidal method for integration. Then, we find:

$$\vec{y}_{k+1} = \vec{y}_k + \frac{h}{2}(F[\vec{y}_k] + F[\vec{y}_{k+1}]) + O(h^3)$$

This is the formula we wrote for the trapezoidal method in §13.3.3.

If we do not wish to solve for $\vec{y}_{k+1}$ implicitly, however, we must find an expression to approximate $F[\vec{y}_{k+1}]$. Using Euler's method, however, we know that $\vec{y}_{k+1} = \vec{y}_k + hF[\vec{y}_k] + O(h^2)$. Making this substitution for $\vec{y}_{k+1}$ does not affect the order of approximation of the trapezoidal time step above, so we can write:

$$\vec{y}_{k+1} = \vec{y}_k + \frac{h}{2}(F[\vec{y}_k] + F[\vec{y}_k + hF[\vec{y}_k]]) + O(h^3)$$

Ignoring the $O(h^3)$ terms yields a new integration strategy known as *Heun's method*, which is second-order accurate and explicit.

If we study stability behavior of Heun's method for $y' = ay$ for $a < 0$, we know:

$$\begin{aligned} y_{k+1} &= y_k + \frac{h}{2}(ay_k + a(y_k + hay_k)) \\ &= \left(1 + \frac{h}{2}a(2 + ha)\right) y_k \\ &= \left(1 + ha + \frac{1}{2}h^2a^2\right) y_k \end{aligned}$$

Thus, the method is stable when

$$\begin{aligned} & -1 \leq 1 + ha + \frac{1}{2}h^2a^2 \leq 1 \\ \iff & -4 \leq 2ha + h^2a^2 \leq 0 \end{aligned}$$

The inequality on the right shows $h \leq \frac{2}{|a|}$, and the one on the left is always true for $h > 0$ and $a < 0$, so the stability condition is $h \leq \frac{2}{|a|}$.

Heun's method is an example of a *Runge-Kutta* method derived by applying quadrature methods to the integral above and substituting Euler steps into $F[\cdot]$. Forward Euler is a first-order accurate Runge-Kutta method, and Heun's method is second-order. A popular fourth-order Runge-Kutta method (abbreviated "RK4") is given by:

$$\begin{aligned} \vec{y}_{k+1} &= \vec{y}_k + \frac{h}{6}(\vec{k}_1 + 2\vec{k}_2 + 2\vec{k}_3 + \vec{k}_4) \\ \text{where } \vec{k}_1 &= F\left[\vec{y}_k\right] \\ \vec{k}_2 &= F\left[\vec{y}_k + \frac{1}{2}h\vec{k}_1\right] \\ \vec{k}_3 &= F\left[\vec{y}_k + \frac{1}{2}h\vec{k}_2\right] \\ \vec{k}_4 &= F\left[\vec{y}_k + h\vec{k}_3\right] \end{aligned}$$

This method can be derived by applying Simpson's rule for quadrature.

Runge-Kutta methods are popular because they are explicit and thus easy to evaluate while providing high degrees of accuracy. The cost of this accuracy, however, is that $F[\cdot]$ must be evaluated more times. Furthermore, Runge-Kutta strategies can be extended to implicit methods that can solve stiff equations.

13.3.5 Exponential Integrators

One class of integrators that achieves strong accuracy when $F[\cdot]$ is approximately linear is to use our solution to the model equation explicitly. In particular, if we were solving the ODE $\vec{y}' = A\vec{y}$, using eigenvectors of A (or any other method) we could find an explicit solution $\vec{y}(t)$ as explained in §13.2.3. We usually write $\vec{y}_{k+1} = e^{Ah}\vec{y}_k$, where e^{Ah} encodes our exponentiation of the eigenvalues (in fact we can find a matrix e^{Ah} from this expression that solves the ODE to time h).

Now, if we write

$$\vec{y}' = A\vec{y} + G[\vec{y}],$$

where G is a nonlinear but small function, we can achieve fairly high accuracy by integrating the A part explicitly and then approximating the nonlinear G part separately. For example, the *first-order exponential integrator* applies forward Euler to the nonlinear G term:

$$\vec{y}_{k+1} = e^{Ah}\vec{y}_k - A^{-1}(1 - e^{Ah})G[\vec{y}_k]$$

Analysis revealing the advantages of this method is more complex than what we have written, but intuitively it is clear that these methods will behave particularly well when G is small.

13.4 Multivalue Methods

The transformations in §13.2.1 enabled us to simplify notation in the previous section considerably by reducing all explicit ODEs to the form $\vec{y}' = F[\vec{y}]$. In fact, while all explicit ODEs *can* be written this way, it is not clear that they always *should*.

In particular, when we reduced k-th order ODEs to first-order ODEs, we introduced a number of variables representing the first through $k-1$-st derivatives of the desired output. In fact, in our final solution we only care about the zeroth derivative, that is, the function itself, so orders of accuracy on the temporary variables are less important.

From this perspective, consider the Taylor series

$$\vec{y}(t_k + h) = \vec{y}(t_k) + h\vec{y}'(t_k) + \frac{h^2}{2}\vec{y}''(t_k) + O(h^3)$$

If we only know $\vec{y}'$ up to $O(h^2)$, this does not affect our approximation, since $\vec{y}'$ gets multiplied by h. Similarly, if we only know $\vec{y}''$ up to $O(h)$, this approximation will not affect the Taylor series terms above because it will get multiplied by $h^2/2$. Thus, we now consider "multivalue" methods, designed to integrate $\vec{y}^{(k)}(t) = F[t, \vec{y}'(t), \vec{y}''(t), \ldots, \vec{y}^{(k-1)}(t)]$ with different-order accuracy for different derivatives of the function $\vec{y}$.

Given the importance of Newton's second law $F = ma$, we will restrict to the case $\vec{y}'' = F[t, \vec{y}, \vec{y}']$; many extensions exist for the less common k-th order case. We introduce a "velocity" vector $\vec{v}(t) = \vec{y}'(t)$ and an "acceleration" vector $\vec{a}$. By our previous reduction, we wish to solve the following first-order system:

$$\begin{aligned}\vec{y}'(t) &= \vec{v}(t)\\ \vec{v}'(t) &= \vec{a}(t)\\ \vec{a}(t) &= F[t, \vec{y}(t), \vec{v}(t)]\end{aligned}$$

Our goal is to derive an integrator specifically tailored to this system.

13.4.1 Newmark Schemes

We begin by deriving the famous class of *Newmark* integrators.[1] Denote $\vec{y}_k$, $\vec{v}_k$, and $\vec{a}_k$ as the position, velocity, and acceleration vectors at time t_k; our goal is to advance to time $t_{k+1} \equiv t_k + h$.

[1] We follow the development in `http://www.stanford.edu/group/frg/course_work/AA242B/CA-AA242B-Ch7.pdf`.

Use $\vec{y}(t)$, $\vec{v}(t)$, and $\vec{a}(t)$ to denote the functions of time assuming we start at t_k. Then, obviously we can write

$$\vec{v}_{k+1} = \vec{v}_k + \int_{t_k}^{t_{k+1}} \vec{a}(t)\, dt$$

We also can write $\vec{y}_{k+1}$ as an integral involving $\vec{a}(t)$, by following a few steps:

$$\begin{aligned}
\vec{y}_{k+1} &= \vec{y}_k + \int_{t_k}^{t_{k+1}} \vec{v}(t)\, dt \\
&= \vec{y}_k + [t\vec{v}(t)]_{t_k}^{t_{k+1}} - \int_{t_k}^{t_{k+1}} t\vec{a}(t)\, dt \text{ after integration by parts} \\
&= \vec{y}_k + t_{k+1}\vec{v}_{k+1} - t_k\vec{v}_k - \int_{t_k}^{t_{k+1}} t\vec{a}(t)\, dt \text{ by expanding the difference term} \\
&= \vec{y}_k + h\vec{v}_k + t_{k+1}\vec{v}_{k+1} - t_{k+1}\vec{v}_k - \int_{t_k}^{t_{k+1}} t\vec{a}(t)\, dt \text{ by adding and subtracting } h\vec{v}_k \\
&= \vec{y}_k + h\vec{v}_k + t_{k+1}(\vec{v}_{k+1} - \vec{v}_k) - \int_{t_k}^{t_{k+1}} t\vec{a}(t)\, dt \text{ after factoring} \\
&= \vec{y}_k + h\vec{v}_k + t_{k+1}\int_{t_k}^{t_{k+1}} \vec{a}(t)\, dt - \int_{t_k}^{t_{k+1}} t\vec{a}(t)\, dt \text{ since } \vec{v}'(t) = \vec{a}(t) \\
&= \vec{y}_k + h\vec{v}_k + \int_{t_k}^{t_{k+1}} (t_{k+1} - t)\vec{a}(t)\, dt
\end{aligned}$$

Suppose we choose $\tau \in [t_k, t_{k+1}]$. Then, we can write expressions for $\vec{a}_k$ and $\vec{a}_{k+1}$ using the Taylor series about τ:

$$\begin{aligned}
\vec{a}_k &= \vec{a}(\tau) + \vec{a}'(\tau)(t_k - \tau) + O(h^2) \\
\vec{a}_{k+1} &= \vec{a}(\tau) + \vec{a}'(\tau)(t_{k+1} - \tau) + O(h^2)
\end{aligned}$$

For any constant $\gamma \in \mathbb{R}$, if we scale the first equation by $1 - \gamma$ and the second by γ and sum the results, we find:

$$\begin{aligned}
\vec{a}(\tau) &= (1-\gamma)\vec{a}_k + \gamma\vec{a}_{k+1} + \vec{a}'(\tau)((\gamma - 1)(t_k - \tau) - \gamma(t_{k+1} - \tau)) + O(h^2) \\
&= (1-\gamma)\vec{a}_k + \gamma\vec{a}_{k+1} + \vec{a}'(\tau)(\tau - h\gamma - t_k) + O(h^2) \text{ after substituting } t_{k+1} = t_k + h
\end{aligned}$$

In the end, we wish to integrate $\vec{a}$ from t_k to t_{k+1} to get the change in velocity. Thus, we compute:

$$\begin{aligned}
\int_{t_k}^{t_{k+1}} \vec{a}(\tau)\, d\tau &= (1-\gamma)h\vec{a}_k + \gamma h\vec{a}_{k+1} + \int_{t_k}^{t_{k+1}} \vec{a}'(\tau)(\tau - h\gamma - t_k)\, d\tau + O(h^3) \\
&= (1-\gamma)h\vec{a}_k + \gamma h\vec{a}_{k+1} + O(h^2),
\end{aligned}$$

where the second step holds because the integrand is $O(h)$ and the interval of integration is of width h. In other words, we now know:

$$\vec{v}_{k+1} = \vec{v}_k + (1-\gamma)h\vec{a}_k + \gamma h\vec{a}_{k+1} + O(h^2)$$

To make a similar approximation for $\vec{y}_{k+1}$, we can write

$$\begin{aligned}
\int_{t_k}^{t_{k+1}} (t_{k+1} - t)\vec{a}(t)\, dt &= \int_{t_k}^{t_{k+1}} (t_{k+1} - \tau)((1-\gamma)\vec{a}_k + \gamma\vec{a}_{k+1} + \vec{a}'(\tau)(\tau - h\gamma - t_k))\, d\tau + O(h^3) \\
&= \frac{1}{2}(1-\gamma)h^2\vec{a}_k + \frac{1}{2}\gamma h^2\vec{a}_{k+1} + O(h^2) \text{ by a similar argument.}
\end{aligned}$$

Thus, we can use our earlier relationship to show:

$$\begin{aligned}\vec{y}_{k+1} &= \vec{y}_k + h\vec{v}_k + \int_{t_k}^{t_{k+1}} (t_{k+1} - t)\vec{a}(t)\, dt \text{ from before} \\ &= \vec{y}_k + h\vec{v}_k + \left(\frac{1}{2} - \beta\right) h^2\vec{a}_k + \beta h^2\vec{a}_{k+1} + O(h^2)\end{aligned}$$

Here, we use β instead of γ (and absorb a factor of two in the process) because the γ we choose for approximating $\vec{y}_{k+1}$ does not have to be the same as the one we choose for approximating $\vec{v}_{k+1}$.

After all this integration, we have derived the class of Newmark schemes, with two input parameters γ and β, which has first-order accuracy by the proof above:

$$\begin{aligned}\vec{y}_{k+1} &= \vec{y}_k + h\vec{v}_k + \left(\frac{1}{2} - \beta\right) h^2\vec{a}_k + \beta h^2\vec{a}_{k+1} \\ \vec{v}_{k+1} &= \vec{v}_k + (1 - \gamma)h\vec{a}_k + \gamma h\vec{a}_{k+1} \\ \vec{a}_k &= F[t_k, \vec{y}_k, \vec{v}_k]\end{aligned}$$

Different choices of β and γ lead to different schemes. For instance, consider the following examples:

- $\beta = \gamma = 0$ gives the *constant acceleration* integrator:

  $$\begin{aligned}\vec{y}_{k+1} &= \vec{y}_k + h\vec{v}_k + \frac{1}{2}h^2\vec{a}_k \\ \vec{v}_{k+1} &= \vec{v}_k + h\vec{a}_k\end{aligned}$$

  This integrator is explicit and holds exactly when the acceleration is a constant function.

- $\beta = 1/2, \gamma = 1$ gives the *constant implicit acceleration* integrator:

  $$\begin{aligned}\vec{y}_{k+1} &= \vec{y}_k + h\vec{v}_k + \frac{1}{2}h^2\vec{a}_{k+1} \\ \vec{v}_{k+1} &= \vec{v}_k + h\vec{a}_{k+1}\end{aligned}$$

  Here, the velocity is stepped implicitly using backward Euler, giving first-order accuracy. The update of $\vec{y}$, however, can be written

  $$\vec{y}_{k+1} = \vec{y}_k + \frac{1}{2}h(\vec{v}_k + \vec{v}_{k+1}),$$

  showing that it locally is updated by the midpoint rule; this is our first example of a scheme where the velocity and position updates have different orders of accuracy. Even so, it is possible to show that this technique, however, is globally first-order accurate in $\vec{y}$.

- $\beta = 1/4, \gamma = 1/2$ gives the following second-order trapezoidal scheme after some algebra:

  $$\begin{aligned}\vec{x}_{k+1} &= \vec{x}_k + \frac{1}{2}h(\vec{v}_k + \vec{v}_{k+1}) \\ \vec{v}_{k+1} &= \vec{v}_k + \frac{1}{2}h(\vec{a}_k + \vec{a}_{k+1})\end{aligned}$$

- $\beta = 0, \gamma = 1/2$ gives a second-order accurate *central differencing* scheme. In the canonical form, we have

$$\vec{x}_{k+1} = \vec{x}_k + h\vec{v}_k + \frac{1}{2}h^2\vec{a}_k$$
$$\vec{v}_{k+1} = \vec{v}_k + \frac{1}{2}h(\vec{a}_k + \vec{a}_{k+1})$$

  The method earns its name because simplifying the equations above leads to the alternative form:

$$\vec{v}_{k+1} = \frac{\vec{y}_{k+2} - \vec{y}_k}{2h}$$
$$\vec{a}_{k+1} = \frac{\vec{y}_{k+1} - 2\vec{y}_{k+1} + \vec{y}_k}{h^2}$$

It is possible to show that Newmark's methods are unconditionally stable when $4\beta > 2\gamma > 1$ and that second-order accuracy occurs exactly when $\gamma = 1/2$ (CHECK).

13.4.2 Staggered Grid

A different way to achieve second-order accuracy in $\vec{y}$ is to use centered differences about the time $t_{k+1/2} \equiv t_k + h/2$:

$$\vec{y}_{k+1} = \vec{y}_k + h\vec{v}_{k+1/2}$$

Rather than use Taylor arguments to try to move $\vec{v}_{k+1/2}$, we can simply store velocities $\vec{v}$ at *half* points on the grid of time steps.

Then, we can use a similar update to step forward the velocities:

$$\vec{v}_{k+3/2} = \vec{v}_{k+1/2} + h\vec{a}_{k+1}.$$

Notice that this update actually is second-order accurate for $\vec{x}$ as well, since if we substitute our expressions for $\vec{v}_{k+1/2}$ and $\vec{v}_{k+3/2}$ we can write:

$$\vec{a}_{k+1} = \frac{1}{h^2}(\vec{y}_{k+2} - 2\vec{y}_{k+1} + \vec{y}_k)$$

Finally, a simple approximation suffices for the acceleration term since it is a higher-order term:

$$\vec{a}_{k+1} = F\left[t_{k+1}, \vec{x}_{k+1}, \frac{1}{2}(\vec{v}_{k+1/2} + \vec{v}_{k+3/2})\right]$$

This expression can be substituted into the expression for $\vec{v}_{k+3/2}$.

When $F[\cdot]$ has no dependence on $\vec{v}$, the method is fully explicit:

$$\vec{y}_{k+1} = \vec{y}_k + h\vec{v}_{k+1/2}$$
$$\vec{a}_{k+1} = F[t_{k+1}, \vec{y}_{k+1}]$$
$$\vec{v}_{k+3/2} = \vec{v}_{k+1/2} + h\vec{a}_{k+1}$$

This is known as the *leapfrog* method of integration, thanks to the staggered grid of times and the fact that each midpoint is used to update the next velocity or position.

Otherwise, if the velocity update has dependence on $\vec{v}$ then the method becomes implicit. Often times, dependence on velocity is symmetric; for instance, wind resistance simply changes sign if you reverse the direction you are moving. This property can lead to symmetric matrices in the implicit step for updating velocities, making it possible to use conjugate gradients and related fast iterative methods to solve.

13.5 To Do

- Define stiff ODE
- Give table of time stepping methods for $F[t; \vec{y}]$
- Use $\vec{y}$ notation more consistently

13.6 Problems

- TVD RK
- Multistep/multivalue methods a la Heath
- Verlet integration
- Symplectic integrators